T-404-LOKA
15. December 2014

CCP - Code Management System

Students
Hannes Guðmundsson
Jón Örn Sigurðsson

Instructor
Torfi H. Leifsson

HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Contents

Table of figures 2
Table of tables 2
Introduction 3
Project description 4
Organization 5
Scrum 5
Scrum roles 5
Development environment 5
Testing 6
Progress summary 7
Sprint 1 8
Sprint 2 9
Sprint 3 10
Sprint 4 11
Sprint 5 12
Sprint 6 13
Sprint 7 14
Sprint 8 15
Summary 16

Table of figures

Figure 1 Diagram of the project 4
Figure 2 Burndown chart for sprint 1 8
Figure 3 Burndown chart for sprint 2 9
Figure 4 Burndown chart for sprint 3 10
Figure 5 Burndown chart for sprint 4 11
Figure 6 Burndown chart for sprint 5 12
Figure 7 Burndown chart for sprint 6 13
Figure 8 Burndown chart for sprint 7 14
Figure 9 Burndown chart for sprint 8 15

Table of tables

Table 1 Sprint days 5
Table 2 Scrum roles 5
Table 3 Sprint backlog for sprint 1 8
Table 4 Sprint backlog for sprint 2 9
Table 5 Sprint backlog for sprint 3 10
Table 6 Sprint backlog for sprint 4 11
Table 7 Sprint backlog for sprint 5 12
Table 8 Sprint backlog for sprint 6 13
Table 9 Sprint backlog for sprint 7 14
Table 10 Sprint backlog for sprint 8 15

Introduction

The project was done in collaboration with CCP. The aim of the project was to create a service to handle codes for various intellectual properties at CCP. The codes are either sold by a third-party reseller or given away by CCP on conventions or other events that customers can then use to redeem various items and services.

The main goal of the project was to simplify the process of creating and managing codes, removing the necessity of having CCP developer spending his/hers time on manually creating the codes in current fashion and saving valuable development time for something more relevant for CCP's core business. The service also enables CCP to retire legacy code and management of CD keys and EVE time codes. It also allows sales and marketing at CCP to be self-sufficient in creating and managing codes.

Project description

The project is twofold. On one hand is the code service itself, which is a REST service with two endpoints, one for when a customer validates and redeems a code and another for creating and managing codes. The code service communicates with other services already in place at CCP. On the other hand is the admin pages which is a web application that CCP employees will use to manage codes. The admin pages will interact with the code service to create and manage codes. Even though the project revolves around creating the code service a big chunk of it is designing and implementing the user interface for the admin pages.

The diagram of the project can be seen in figure 1. The part colored green is our part of the project and the part colored blue are services already in place that our system will communicate with.

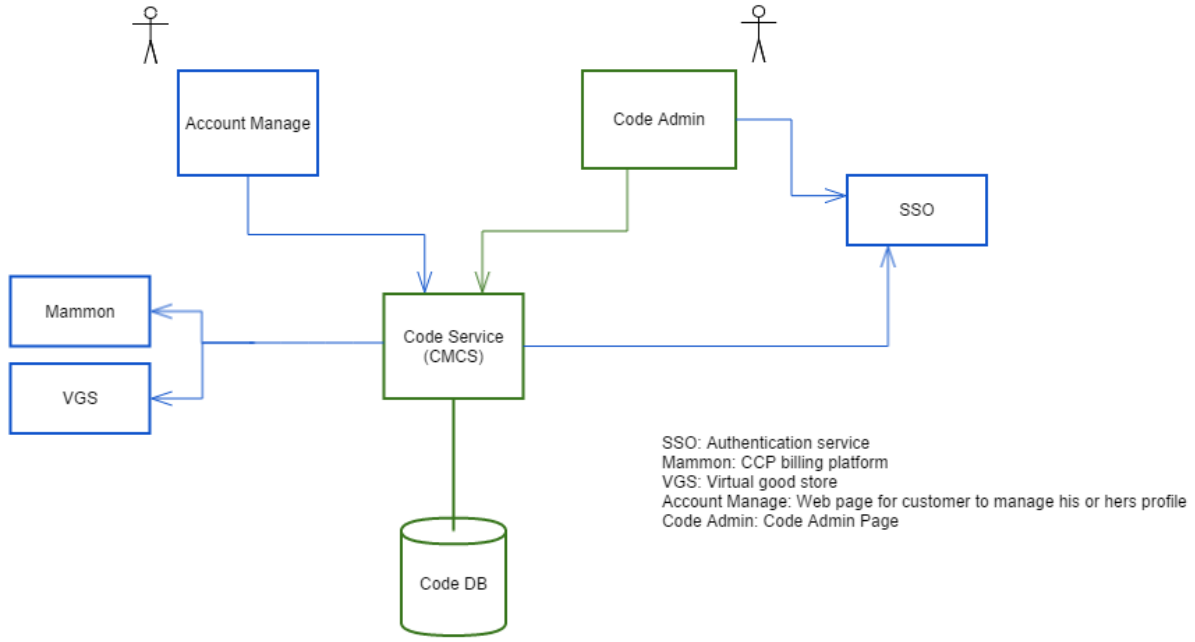

Figure 1 Diagram of the project

Explanations for diagram:

- Mammon is CCP's billing platform
- VGS is a virtual good store
- Account manage is a web page for customer to manage his or hers profile
- Code admin is the UI for creating and managing codes
- SSO is the authentication service

Organization

Scrum

We used scrum to organize our work and used Excel to keep track of the product backlog, sprints and keep track of hours worked. The product backlog was created in collaboration with CCP and the project owner. The team worked either at home or at school and weekly status meetings were held with the project owner at CCP headquarters where the progress of the project was reviewed and the product backlog was prioritized. We used Dropbox to store all our documents, including the product backlog, sprint backlog and reports and the code was stored in a private repository on GitHub. The complete product backlog and sprint backlogs can be found on the project CD.

The sprints were split into 4 two weeks sprints and 4 one week sprints. The first week we used to plan our work and was labelled as sprint zero. We split the sprints up in this fashion because in the end of the period we were able to spend more time on the project. How the sprints were divided up and the duration of each sprint can be seen in table 1.

Sprint	Days
Sprint zero	25. August – 2. September
Sprint 1	3. September – 16. September
Sprint 2	17. September – 30. September
Sprint 3	1. October – 14. October
Sprint 4	15. October – 28. October
Sprint 5	29. October – 5. November
Sprint 6	24. November – 30. November (in this sprint we had the final exams, so there was not much work done in this week)
Sprint 7	1. December – 7. December
Sprint 8	8. December – 14. December

Table 1 Sprint days

Scrum roles

The team divided the roles as can be seen in table 2.

Role	Member
Project owner	Höskuldur Sigurðsson (CCP)
Scrum master	Hannes Guðmundsson
Development team	Hannes Guðmundsson Jón Örn Sigurðsson

Table 2 Scrum roles

Development environment

Visual Studio 2013 was used to program the code service and the admin pages and SQL Management Studio was used to manage the database.

Testing

Integration tests were done for the code service where the system was tested. We decided to only go with integration tests since they test the system as a whole and therefore give a good image if the code is working as it should or not. We decided to do the integration tests thorough and skip unit tests instead. We ended up creating roughly 170 integration tests which covered all endpoints in the code service system.

The integration tests helped us to correct few errors that we had not discovered by using the system normally. This was really helpful since we were able to correct errors that otherwise is not sure we would have spotted.

Progress summary

The project was split up into 4 two week sprints and 4 one week sprints. Most of the sprints went according to plan, except sprint 1, 3 and 4. In sprint one we were getting familiar to the programming environment and in sprint 3 and 4 we both overestimated what we would be able to finish and workload in other courses was taking away a lot of our time.

We were very pleased with the progress of the project and CCP was also pleased with the outcome of the project. We finished all of the A requirements but some A requirements were downgraded to a B requirement during the progress of the project. We spent in total 778 hours on the project but had estimated 728 hours. A detailed summary of hours spent on the project broken down into categories can be found on the project CD.

Sprint 1

As can be seen on the sprint burndown chart we had a bit of a rough start, neither one of us was familiar with asynchronous programming and therefore it took us longer than we had estimated to finish the stories. We didn't really get going until approaching the end of the sprint. We had to move 2 stories into the next sprint.

Id	User story	Priority	Story Points	Status
4	As a CMCS admin I can create a single code type	A	5	Done
7	As a CMCS admin I can view the code types in the admin pages	A	5	Done
42	Create progress summary report 1	A	4	Done
43	Prepare presentation for status meeting 1	A	4	Done
8	As a CMCS admin I can browse through the N number of code pages with paging	A	8	In progress
5	As a CMCS admin I can delete the code type	A	5	Not started

Table 3 Sprint backlog for sprint 1

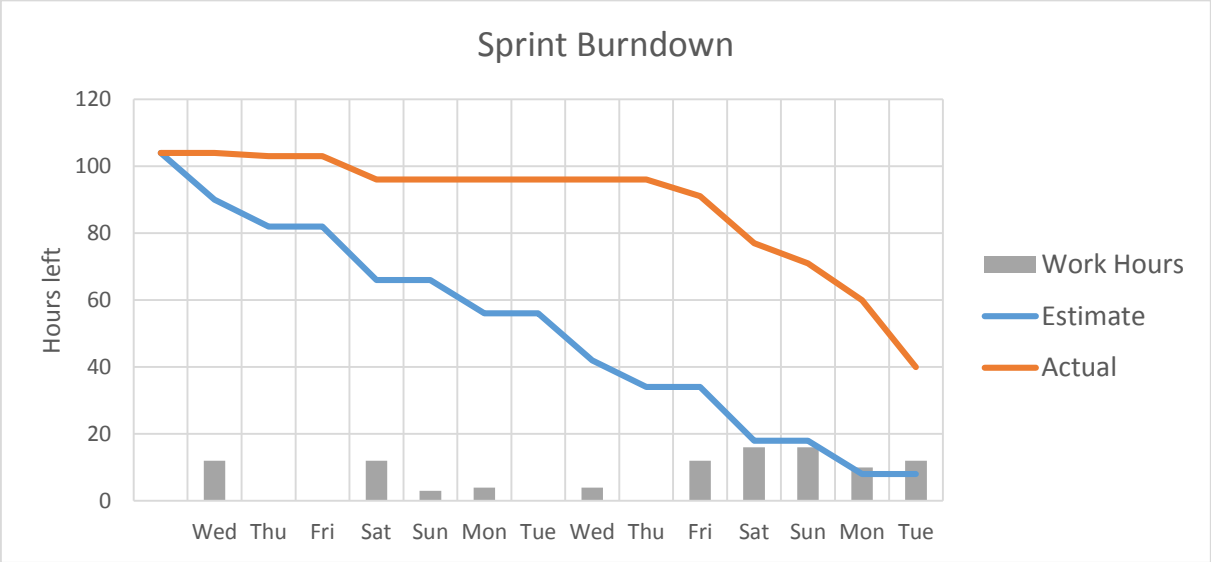

Figure 2 Burndown chart for sprint 1

Sprint 2

Sprint 2 went really well and without any major problems. We were able to finish all the stories we had planned for the sprint before the sprint was over so we had to take in another story from the product backlog.

Id	User story	Priority	Story Points	Status
8	As a CMCS admin I can browse through the N number of code pages with paging	A	8	Done
5	As a CMCS admin I can delete the code type	A	5	Done
6	As a CMCS admin I can edit the code type	A	5	Done
9	As a CMCS admin I can sort the code types based on attributes	A	5	Done
10	As a CMCS admin I can search for a specific code type	A	3	Done
11	As a CMCS admin I can create a Reseller	A	6	Done
12	As a CMCS admin I can edit a Reseller	A	5	Done
13	As a CMCS admin I can delete a Reseller	A	5	Done
14	As a CMCS admin I can create a pool	A	10	Done

Table 4 Sprint backlog for sprint 2

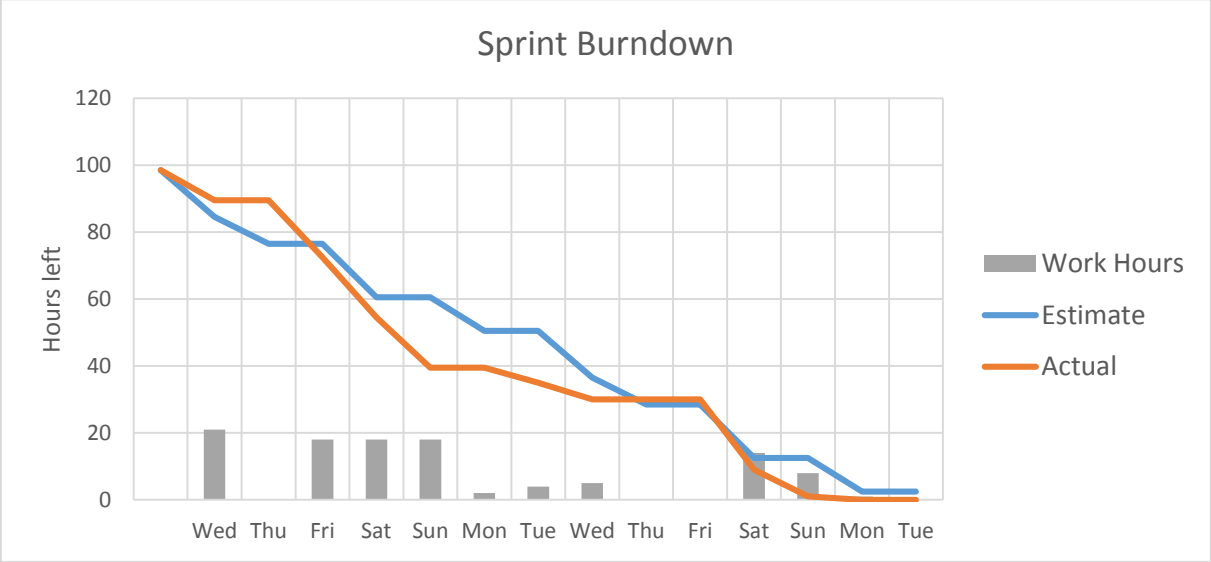

Figure 3 Burndown chart for sprint 2

Sprint 3

In this sprint we did some code cleanup and refactoring, we also added proper exception handling and validation. When we planned the sprint we did not take this into account. CCP asked us if we could send them the code for code review so we decided to spend some time on code cleanup. Therefore we were not able to finish four stories and had to move them to the next sprint.

Id	User story	Priority	Story Points	Status
15	As a CMCS admin I can edit a pool	A	6	Done
16	As a CMCS admin I can expire a pool	A	8	Done
17	EPIC: As a CMCS admin I can search, view, filter, sort for resellers and pools	A	10	Done
18	As a CMCS admin when I create a pool I can create codes	A	8	Done
20	As a CMCS admin I can view the code in the admin pages	A	4	Done
21	As a CMCS admin I can browse through the N number of pages with paging	A	4	Done
19	As a CMCS admin I can expire code or a batch of codes	A	6	Not started
22	As a CMCS admin I can filter the codes	A	5	In progress
23	As a CMCS admin I can group code types together	A	4	In progress
24	As a CMCS admin I can add a product to a code type	A	3	In progress

Table 5 Sprint backlog for sprint 3

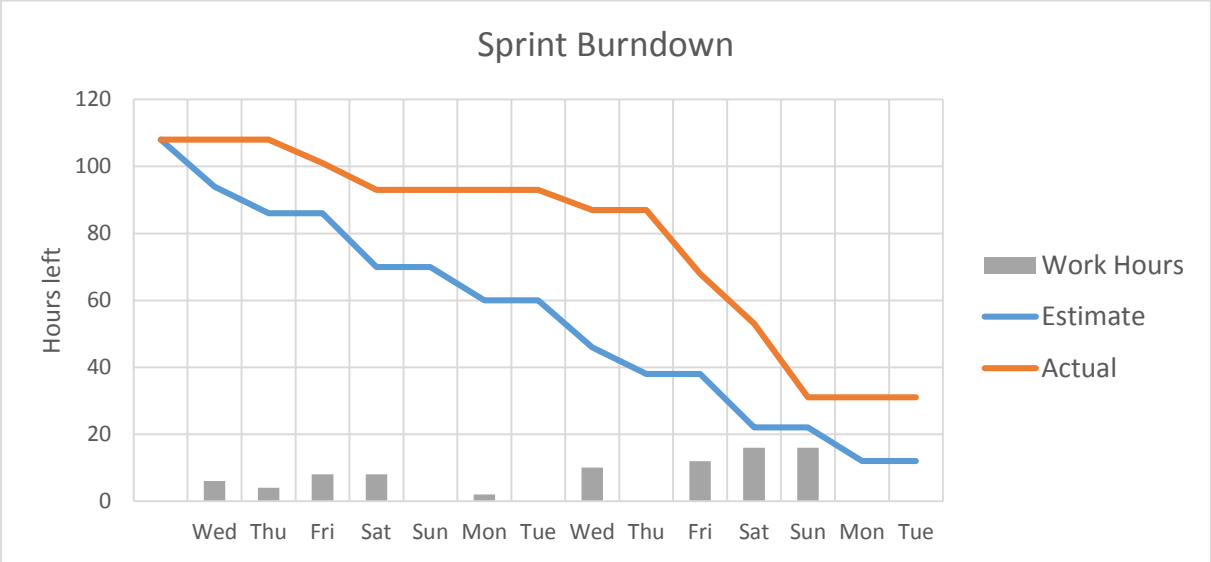

Figure 4 Burndown chart for sprint 3

Sprint 4

This sprint went pretty well. There was only one story we had to move to the next sprint. This sprint was used to finish writing integration tests for the REST service, this was time consuming and took more time than we expected.

Id	User story	Priority	Story Points	Status
19	As a CMCS admin I can expire code or a batch of codes	A	6	Done
22	As a CMCS admin I can filter the codes	A	5	Done
23	As a CMCS admin I can group code types together	A	4	Done
24	As a CMCS admin I can add a product to a code type	A	3	Done
39	Create progress summary report 2	A	4	Done
40	Prepare presentation for status meeting 2	A	4	Done
26	Integration tests for business logic	A	8	Done

Table 6 Sprint backlog for sprint 4

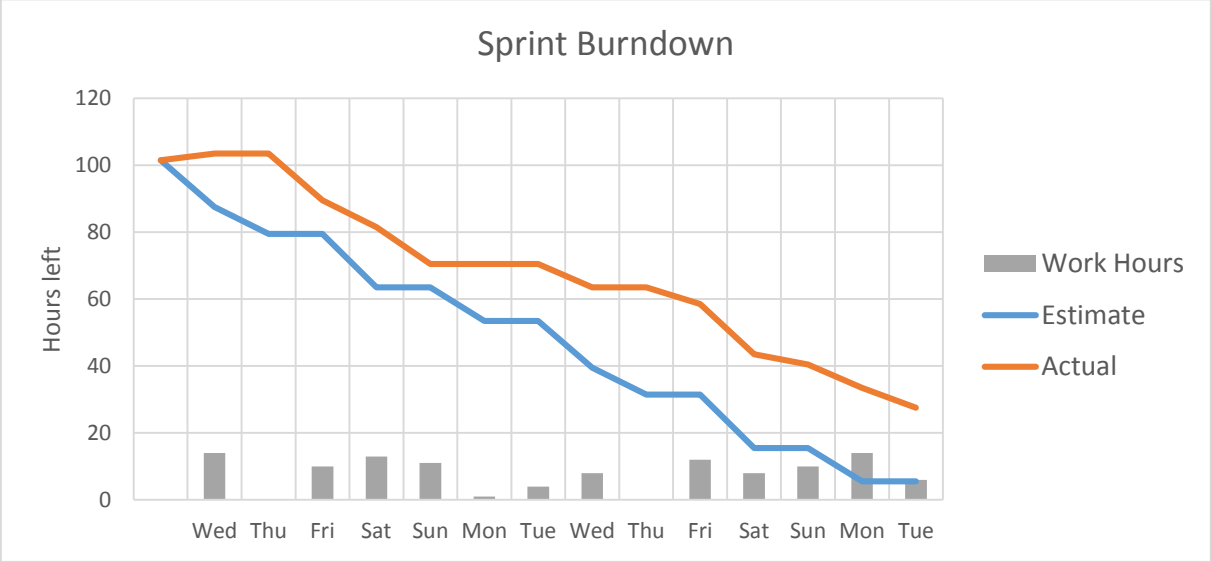

Figure 5 Burndown chart for sprint 4

Sprint 5

This sprint was only a week and Jón Örn was working alone in this sprint because Hannes had heavy workload in other courses therefore we only took 2 stories into this sprint.

Id	User story	Priority	Story Points	Status
25	As a code admin I can export a pool of codes into a text file	A	3	Done
44	As a CMCS admin I can view products that have been added to a code type	A	4	Done

Table 7 Sprint backlog for sprint 5

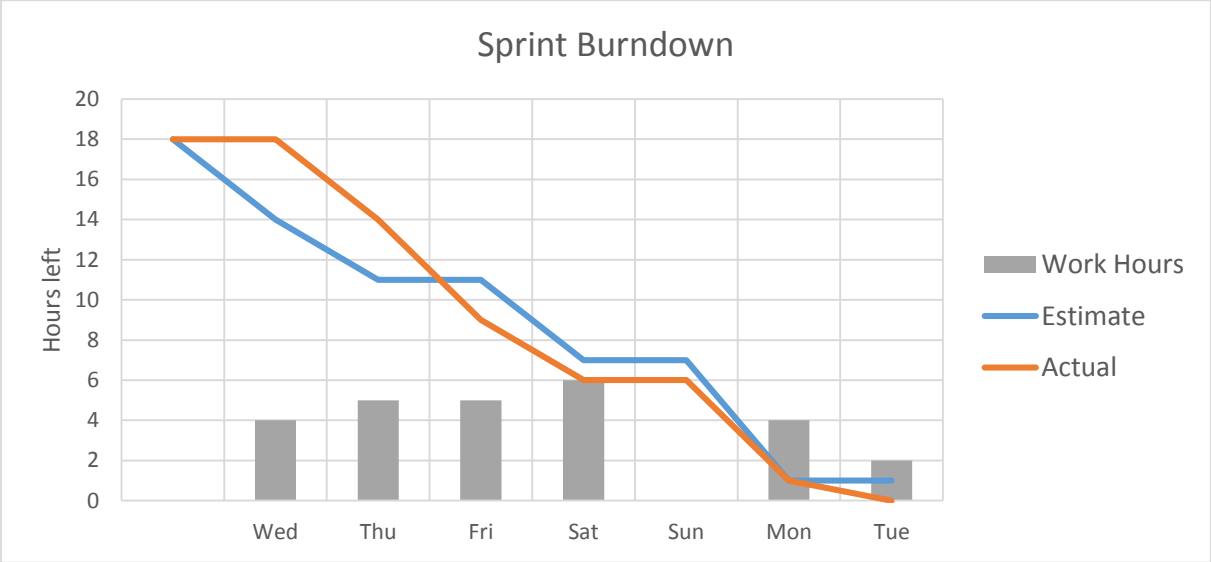

Figure 6 Burndown chart for sprint 5

Sprint 6

We estimated to spend only half the week working on the project but we were able to spend more time than that because the final exams finished earlier than we expected. Although we had more time than planned we underestimated the work that was needed to mock the VGS service.

Id	User story	Priority	Story Points	Status
30	As a customer I can validate a code	A	5	Done
27	As a customer I can redeem a code and have the offers associated with it delivered to my account	A	10	Done
50	Create mock project for mocking connection to the VGS service	A	12	Done

Table 8 Sprint backlog for sprint 6

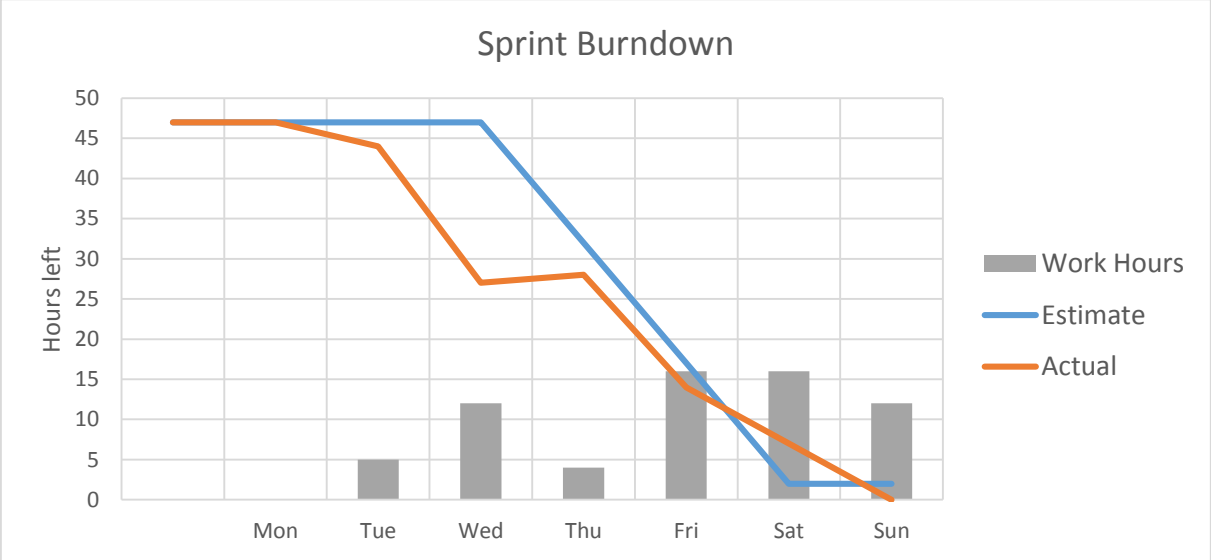

Figure 7 Burndown chart for sprint 6

Sprint 7

This sprint went really well. We were able to finish all stories we planned for the sprint. We worked a few more hours than we had planned which was good because we were few hours short in total hours according to the plan. It took a little more time than we had estimated to create the sale in the VGS store when code is redeemed and deploying the projects also took a little more time than expected.

Id	User story	Priority	Story Points	Status
45	Create the Developers guide	A	6	Done
46	Create the Users guide	A	6	Done
47	Create progress summary report 3	A	4	Done
48	Prepare presentation for status meeting 3	A	6	Done
49	As a CMCS admin I can get service token from SSO	A	10	Done
51	As a customer I can redeem code - Use real VGS service instead of mock	A	6	Done
52	As a code admin I can add products to a code type - Use VGS service instead of mock	A	4	Done
53	As a CMCS admin I can remove a product from a code type	A	4	Done
29	As a customer I can see what is included in the code	A	4	Done
55	Deploy the project on a production server	A	6	Done

Table 9 Sprint backlog for sprint 7

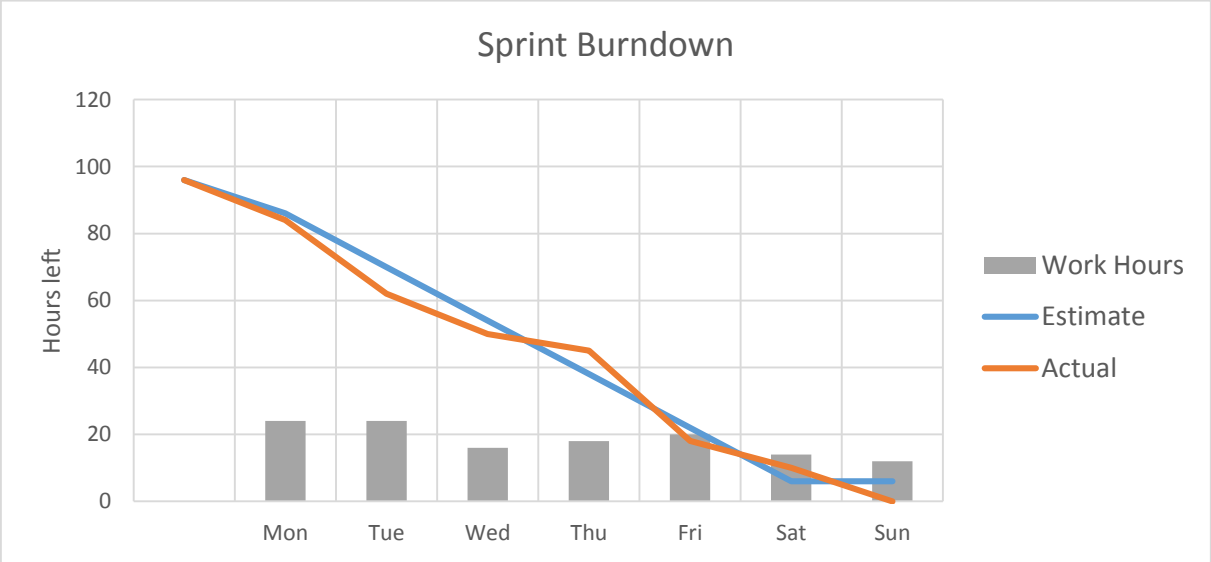

Figure 8 Burndown chart for sprint 7

Sprint 8

In this sprint we were able to finish all stories that we planned. We put in a lot of more hours than we had estimated but it was needed to be able to finish the project. We were running into all kinds of problems when authenticating users. We also had a hard time verifying a token sent in the authorization header to the service project.

Id	User story	Priority	Story Points	Status
33	As a CMCS admin I can login to the system	A	12	Done
55	Create a mock project that validates the code using the REST service	A	8	Done
56	Authorization for the CMCS service project	A	10	Done
57	Create the final report	A	10	Done
58	Deploy the final version of the project on production server	A	6	Done

Table 10 Sprint backlog for sprint 8

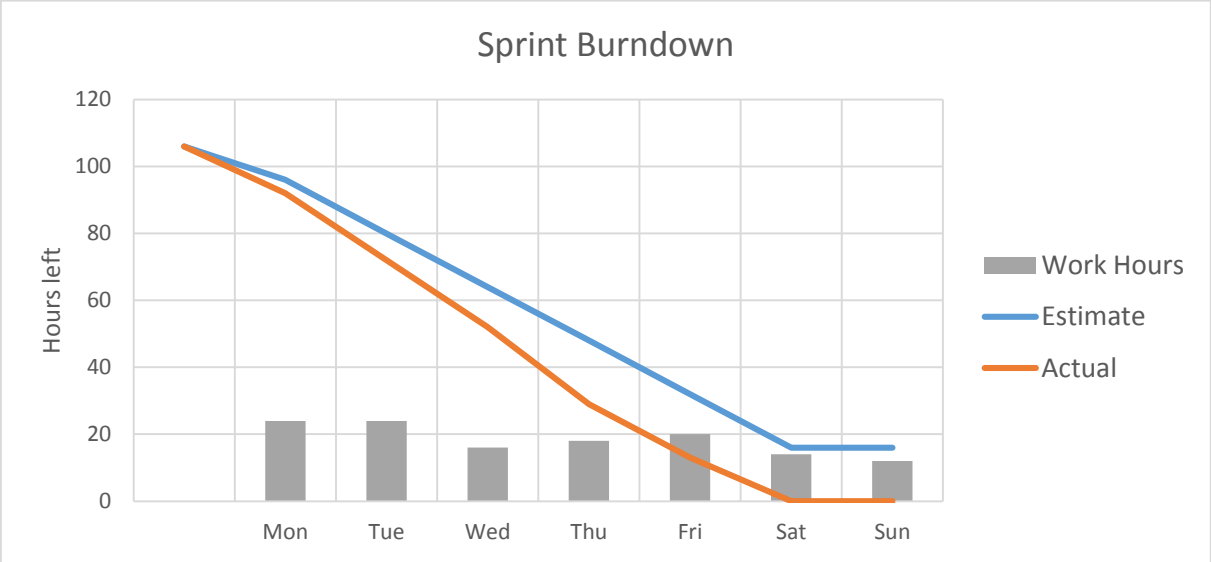

Figure 9 Burndown chart for sprint 8

Summary

We are very pleased that we were able to deliver a working system that is ready for deployment. We finished most of the requirements we had planned for the project. Some of the B requirements that would be nice to have for the system, but not essential for it to function were not finished.

Working on the project was a real good learning experience for the team. What we can take from this project is a deeper understanding of what it is to design and develop an up to date software system. We also got a lot of experience working with asynchronous programming and working in the Microsoft .NET programming environment.

Working with CCP was very pleasant and our contacts there were very helpful and easy to work with. We had weekly meetings at the CCP headquarters with the product owner and programmers that will manage the system. These meetings were a very big factor on how well the project turned out. There we always got very good guidance about how the system was supposed to behave and what would be good idea to work on next in the development of the system.

The code service was implemented using CCP coding convention and system architecture used in systems developed by them. We dedicated ourselves to keeping these principles throughout the project and believe they have taught us a lot about programming a real professional software. Learning asynchronous programming was a rather steep learning curve but as the project progressed we always got more and more comfortable with it.

Our future vision is that the system will be integrated at CCP and will be used by employees to create and manage codes and by customers to redeem codes. We can then hopefully brag to fellow gamers that we made the system they are using to redeem their goods from CCP.