CREST Explorer

Self discovery API explorer

Final report

Spring 2013

Ingi Steinn Guðmundsson
Hjörleifur Henriksson
BSc in Computer Science

Instructor: Stefán Freyr Stefánsson
Examiner: Birgir Kristmannsson

T-404-LOKA
Tölvunarfræðideild

Table of Contents

1. Introduction

CCP recently rolled out a new RESTful service called Carbon REST, or CREST for short. CREST is a service that allows interaction with CCP‘s games EVE and Dust 514 using the HTTP protocol. Quality assurance and developers need to interact with the CREST service today to see what is available to them and what they need to do to call and use a specific service. Interacting with the system today is done through command line tools and even though it is possible to explore the CREST service it can be a cumbersome and time-consuming process, that requires knowledge of the system and good understanding of the HTTP protocol.

The primary goal of the project was to create a new tool for CCP‘s quality assurance and developers to facilitate interactions with CREST in a fast and easy way. We created a self discovery API explorer called CREST Explorer. The system had to be able to create an interface for the user dynamically, depending on what the user is looking at and allow him navigate the CREST service in a fluid manner.

2 .Project description

CREST is a service that allows CCP‘s massively-multiplayer online game EVE and their new online first-person shooting game DUST 514 to interact with each other. Even though the games are vastly different and everything in EVE happens in space while everything in DUST 514 takes place on the ground on planets, they exist in the same universe called New Eden. Thanks to CREST, players from those two games can communicate with each other, trade and even kill each other. CREST works as a bridge between those two games, through which for example when players in EVE want to send a mail to a DUST player, the mail is sent to CREST and it is then relayed to the DUST player. Third-party applications can also be created and connected to CREST allowed access to the EVE and DUST 514 worlds.

The problem is that CREST is extensive and created for computer to computer interactions, making it hard for developers to navigate the service with the tools that exist today and see what is available and what they can do in CREST.

The tools that are being used by CCP developers today are command line tools and can be time consuming and difficult to use in some situations. For one request the user has to get a access token from an single sign-on login service, copy that and create a request to an already known path, see Figure1. The response he gets back can be a huge JSON formatted string that does not show all the wanted information or too much information, see Figure 3. CCP also has extensive auto-generated documentation that shows CREST structure but does not allow for interactions with data. The auto-generated documentation can also be time-consuming to read if you don‘t know exactly what you are looking for, see Figure 3.

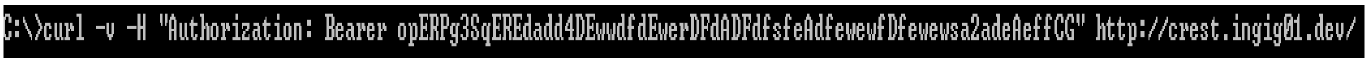
The project aim is to create a system for CCP developers and quality assurance that replaces command line interactions with CREST and serves as an interactive documentation. We call the system CREST Explorer. CREST Explorer is a self discovery API explorer that can create an interface dynamically depending on what service or data the user is looking at, making it much easier to perform testing on and developing for it through immediate explorative access to all exposed branches of the CREST service.

CREST Explorer should be generic enough to handle new services in CREST that are added later. It should also find out what the user can do with specific data. The user can log in as either a DUST514 or EVE character and explore the service as that character to get information, delete, create or change data specific to that character.

As an example of how CREST Explorer could be used, a new service might be made available in CREST. This new service can send in-game mail between players. A developer wanting to use the service for a new feature in the game, instead of going through the documentation, can find it with CREST Explorer. When he selects it, he is presented with all available actions on that service. He chooses to send an in-game mail and a form is generated for him. The form displays the following fields:

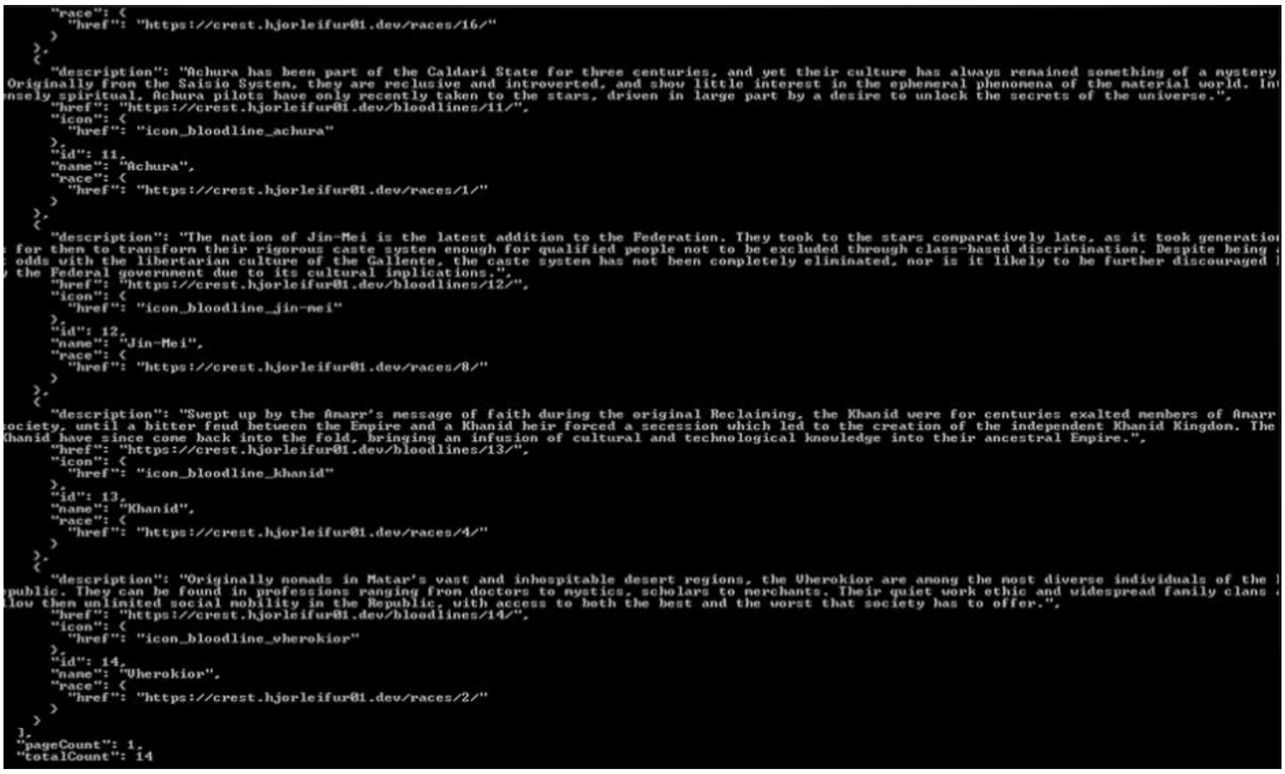
- Subject
- Body
- Recipients

The developer can now see what needs to be entered to send a mail and even test the service before he uses it by filling out the form and submitting it. Quality assurance can also use it to test the new service and see if it is working correctly the same way.



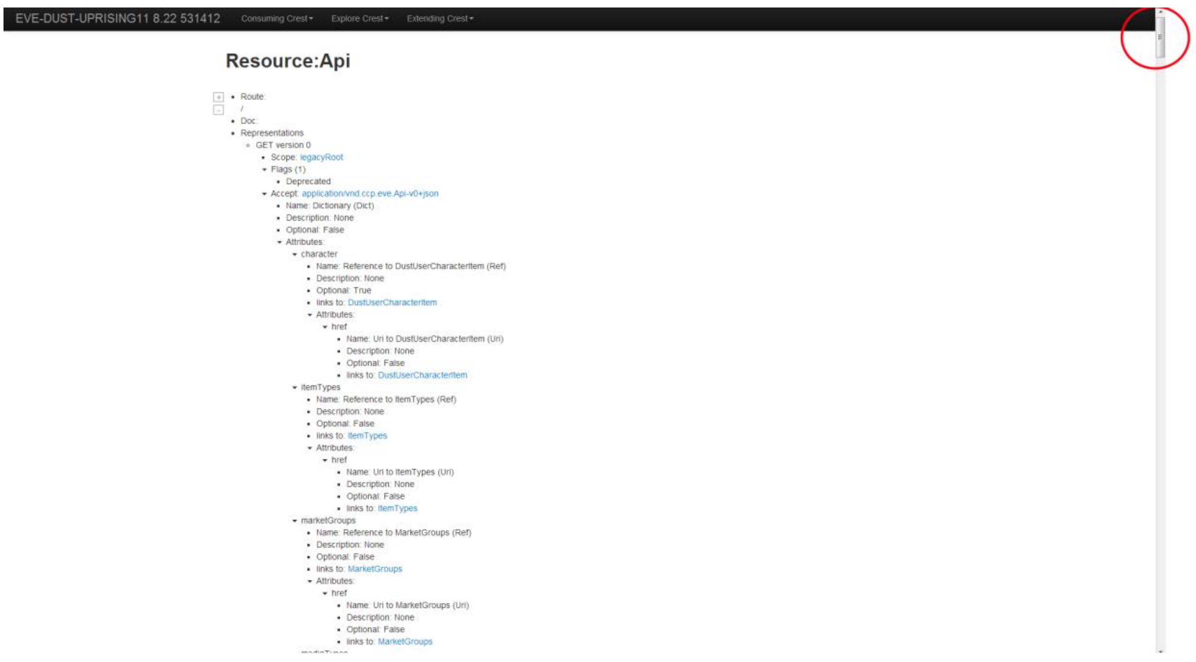
```
C:\>curl -v -H "Authorization: Bearer opERPg3SqEREdadd4DEwwdfdEwerDFdADFdfsfeAdfewewfDfewewsa2adeAeffCG" http://crest.ingig01.dev/
```

Figure 1: Command line request with a single sign-on access token in the header and the wanted path



```
      "race": {
        "href": "https://crest.hjorleifur01.dev/races/16/"
      }
    },
    {
      "description": "Achura has been part of the Caldari State for three centuries, and yet their culture has always remained something of a mystery
 Originally from the Saisio System, they are reclusive and introverted, and show little interest in the ephemeral phenomena of the material world. Int
nsely spiritual, Achura pilots have only recently taken to the stars, driven in large part by a desire to unlock the secrets of the universe.",
      "href": "https://crest.hjorleifur01.dev/bloodlines/11/",
      "icon": {
        "href": "icon_bloodline_achura"
      },
      "id": 11,
      "name": "Achura",
      "race": {
        "href": "https://crest.hjorleifur01.dev/races/1/"
      }
    },
    {
      "description": "The nation of Jin-Mei is the latest addition to the Federation. They took to the stars comparatively late, as it took generatio
 for them to transform their rigorous caste system enough for qualified people not to be excluded through class-based discrimination. Despite being
 odds with the libertarian culture of the Gallente, the caste system has not been completely eliminated, nor is it likely to be further discouraged
 the Federal government due to its cultural implications.",
      "href": "https://crest.hjorleifur01.dev/bloodlines/12/",
      "icon": {
        "href": "icon_bloodline_jin-mei"
      },
      "id": 12,
      "name": "Jin-Mei",
      "race": {
        "href": "https://crest.hjorleifur01.dev/races/8/"
      }
    },
    {
      "description": "Swept up by the Amarr's message of faith during the original Reclaiming, the Khanid were for centuries exalted members of Amarr
ociety, until a bitter feud between the Empire and a Khanid heir forced a secession which led to the creation of the independent Khanid Kingdom. The
hanid have since come back into the fold, bringing an infusion of cultural and technological knowledge into their ancestral Empire.",
      "href": "https://crest.hjorleifur01.dev/bloodlines/13/",
      "icon": {
        "href": "icon_bloodline_khanid"
      },
      "id": 13,
      "name": "Khanid",
      "race": {
        "href": "https://crest.hjorleifur01.dev/races/4/"
      }
    },
    {
      "description": "Originally nomads in Matar's vast and inhospitable desert regions, the Vherokior are among the most diverse individuals of the
public. They can be found in professions ranging from doctors to mystics, scholars to merchants. Their quiet work ethic and widespread family clans
low them unlimited social mobility in the Republic, with access to both the best and the worst that society has to offer.",
      "href": "https://crest.hjorleifur01.dev/bloodlines/14/",
      "icon": {
        "href": "icon_bloodline_vherokior"
      },
      "id": 14,
      "name": "Vherokior",
      "race": {
        "href": "https://crest.hjorleifur01.dev/races/2/"
      }
    }
  ],
  "pageCount": 1,
  "totalCount": 14
```

Figure 2: Command line response



Resource:Api

```
• Route:
    /
• Doc:
• Representations
    ◦ GET version 0
        • Scope: legacyRoot
        ▾ Flags (1)
            • Deprecated
        ▾ Accept: application/vnd.ccp.eve.Api-v0+json
            • Name: Dictionary (Dict)
            • Description: None
            • Optional: False
            ▾ Attributes:
                ▾ character
                    • Name: Reference to DustUserCharacterItem (Ref)
                    • Description: None
                    • Optional: True
                    • links to: DustUserCharacterItem
                    ▾ Attributes:
                        ▾ href
                            • Name: Uri to DustUserCharacterItem (Uri)
                            • Description: None
                            • Optional: False
                            • links to: DustUserCharacterItem
                ▾ itemTypes
                    • Name: Reference to ItemTypes (Ref)
                    • Description: None
                    • Optional: False
                    • links to: ItemTypes
                    ▾ Attributes:
                        ▾ href
                            • Name: Uri to ItemTypes (Uri)
                            • Description: None
                            • Optional: False
                            • links to: ItemTypes
                ▾ marketGroups
                    • Name: Reference to MarketGroups (Ref)
                    • Description: None
                    • Optional: False
                    • links to: MarketGroups
                    ▾ Attributes:
                        ▾ href
                            • Name: Uri to MarketGroups (Uri)
                            • Description: None
                            • Optional: False
                            • links to: MarketGroups
```

Figure 3: Auto-generated documentation for CREST. This image represents one service in CREST and all available options. Note how long the list is marked with the red circle.

3. Approach

3.1 Methodology

The team decided to use SCRUM for this project since CCP makes heavy use of it for most of their projects.

Because all our work had to be done in-house at CCP and we didn‘t have access cards to the building, which prevented us from coming there on weekends and evenings, we decided on steady hours so we could be there at the same time and work together.

At first we decided on 1 week sprints, thinking that would be a good approach for our small team but we quickly ran into problems with that when our school workload increased and almost no work could be done in some sprints. We revised our plans after sprint 3 and made the sprints last two weeks and threw away our time plan to compensate for the delay. We also got our access cards to the building and that allowed for much more flexible hours. The only regular hours we planned was the SCRUM retrospective meetings and the sprint planning which we had on the same day. Daily scrum meetings where also conducted either remotely or at CCP when we were both there.

We used Microsoft Excel to keep track of the product and sprint backlogs. We also used Excel for keeping track of our work hours. The sprints can be seen in Table 1.

Sprints	Begin	End
Sprint 0	16.1.2013	11.2.2013
Sprint 1	11.2.2013	18.2.2013
Sprint 2	18.2.2013	25.2.2013
Sprint 3	25.2.2013	4.3.2013
Sprint 4	4.3.2013	18.3.2013
Sprint 5	18.3.2013	1.4.2013
Sprint 6	1.4.2013	15.4.2013
Sprint 7	15.4.2013	29.4.2013
Sprint 8	29.4.2013	13.5.2013

Table 1. Sprints

It took us some time to set up at CCP, getting access to systems and familiarizing ourselves with them since many of them were quite complex. This led us to start experimenting with pair programming so we could share what we had learned, this proved to be very useful and we stuck with it for most of the project.

During sprint planning, each story that was to be done was reviewed and re-evaluated. The story was broken down into tasks and assigned. A story was not finished until every task in that story was completed. It was then tested by the other team member. Code review was done either during work on the story with pair programming or after it was finished.

2.2 Project roles

The team members had following roles

- Ingi Steinn Guðmundsson – Scrum master, programmer
- Hjörleifur Henriksson – Programmer
- Nicolas Tittley – Product owner (CCP)

2.3 Project Analysis

A program flow chart was created to help us understand how the service should work. It proved to be very valuable in getting us on the same page as to how the system should work. The flow chart was revised a couple of times as we got a better understanding of the inner workings of CREST.

The state of the project and what it could do was shown to the product owner (and sometimes other CCP employees) on a regular basis, followed by discussions about improvements and the future of the project, with particular focus on making the application as useful as possible. Some extra features were added as a result of these discussions, such as showing the response headers from the most recent request to CREST, and the response time.

2.4 Project tools and software

The client is a single page application made with AngularJS and jQuery. A Python Flask server serves up the client and acts as a proxy between the client and CREST. Twitter bootstrap and angular.ui is used for styling in addition to our own modifications. The Sublime Text editor was used for all code and markup.

2.5 Testing

When we had a basic client up and running we created basic tests in JavaScript for every function. These tests were primitive but effective for debugging.

System tests were conducted by the team members when a story was finished. Extensive System tests were also conducted at the end of sprint 7 and 8 so bug fixes could be made before the final release.

2.6 Design

Intially we were unsure about the look,and had a very rough outline of how it should be, as we were not yet sure about many details that only came to light as our understanding of the project deepened. This was in part due to the dynamic nature of the data that the application is working with and displaying in a readable manner. We knew that Twitter Bootstrap would be used, as it was already in use for several related systems, and we wanted to have a familiar look and feel to the user interface.

2.7 Risk assessment

Our Risk assessment plan served as a guide for us if something was slowing down the project or stopping it. Only one risk came into fruition, and that was the risk of other courses taking up more time than we planned.

Risk assessment 1-5	Description	Mitigation plan	Impact
5	Unable to conform backend to project requirements	Test early on and figure out what´s needed to fulfill requirements	Delay and possible dropping of requirements
3	Changes in backend	Make the software as robust as possible - try to avoid unnecessary dependencies	Refactorization, project delay

3	Test server goes down during presentation	Make recordings of application in use, coordinate with CCP	Unable to show software being used live, less interesting presentation
4	Too much time consumed by other courses	Make up for lost time during weekends	Project delay

3. Project progress

Sprint 0 was our longest sprint, we used that time to get set up at CCP, we got our own workstation and go introduced to the team we would be working for. We also got an extensive tutorial on the inner workings of CREST and most of the systems we would be working with. This was much to take in and we used most of sprint 0 to get familiar with the systems, getting and exploring CCP‘s source code.

We started with a project plan that used 1 week sprints,totaling 13 sprints. It soon became obvious to us that it wasn‘t working because of increased workload from other courses. We did account for that in our risk assesment, and to combat that we planned to work on weekends but we did not get access cards to the CCP building fast enough and our sprints suffered. We increased our sprint durations to two weeks so sprints wouldn‘t go to waste when we had a busy week and when we got access to the building our sprints began to go more smoothly.

In the end one A story was left and two B stories.

3.1 Worked hours

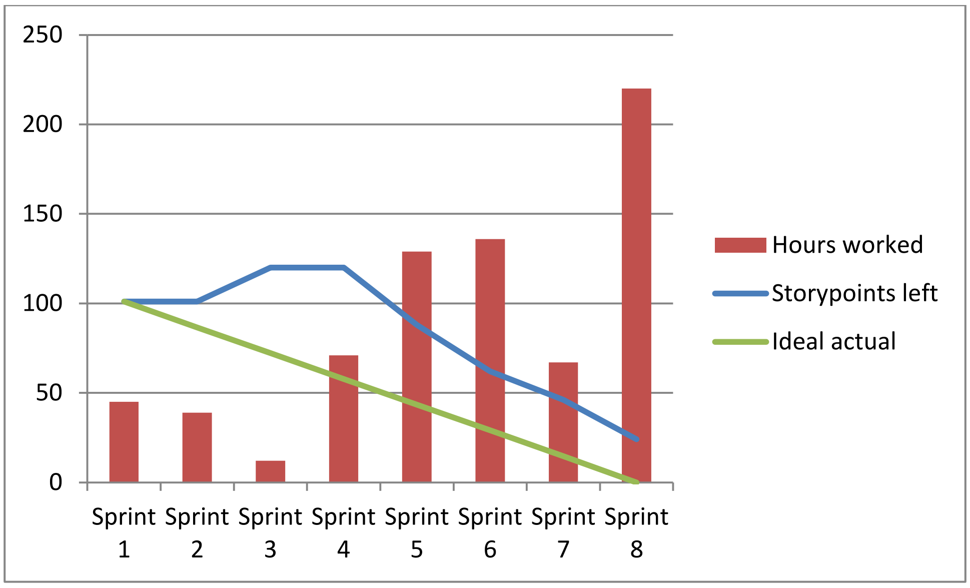
The team members schedule did differ since we did not have all the same courses but in the end we spent almost the same amount of time on the project.

We planned 350 hours per person at the beginning, that estimed was off by 50 hours, see Table 2.

	Ingi	Hjörleifur
Total hours	400	399
Planned hours	350	350

Table 2: Worked hours vs Planned hours

3.2 Release burndown



3.3 Sprint 0

16.01.13 - 11.02.13

Retrospective

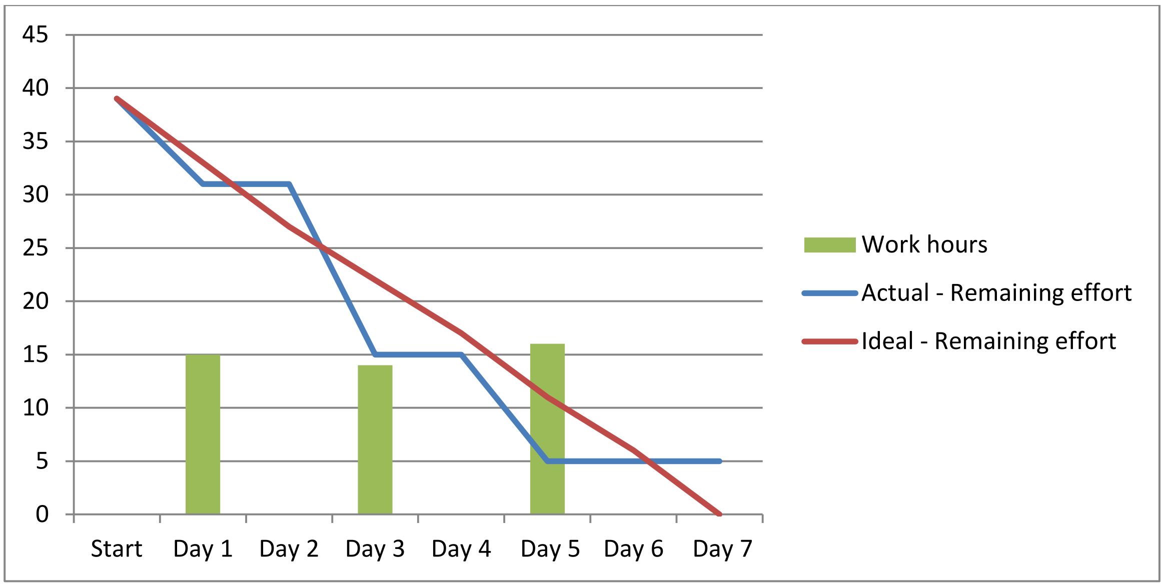
In the initial sprint we spent all of the time doing preparatory work, and getting ourselves set up at the company. This proved a more challenging task than expected, however we gained familiarity with some aspects of the project in the meantime, as we read the documentation and examined some of the source code for the back-end services.

3.4 Sprint 1

11.02.13 - 18.02.13

Retrospective

We had some problems where we had to refactor the estimated time taken for tasks, as the authentication process proved troublesome. The story remains in progress, as we remain undecided about the validation step, so we’ve put that on hold until it becomes clear exactly what is necessary.

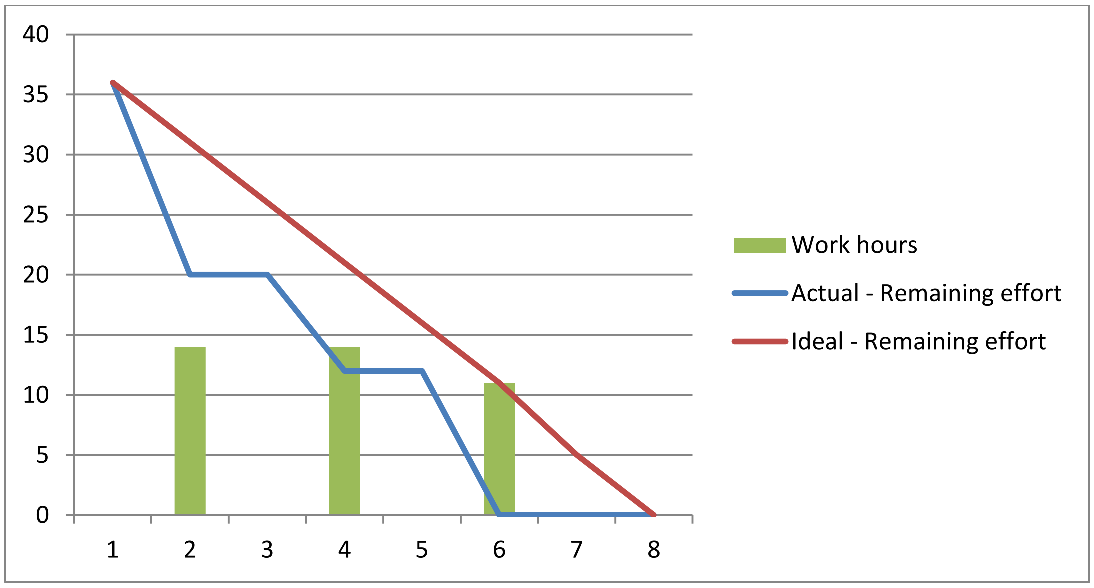


3.5 Sprint 2

18.02.13 - 25.02.13

Retrospective

We put some of the tasks on hold here – getting the available actions requires some backend changes that we will start thinking about in the next sprint.

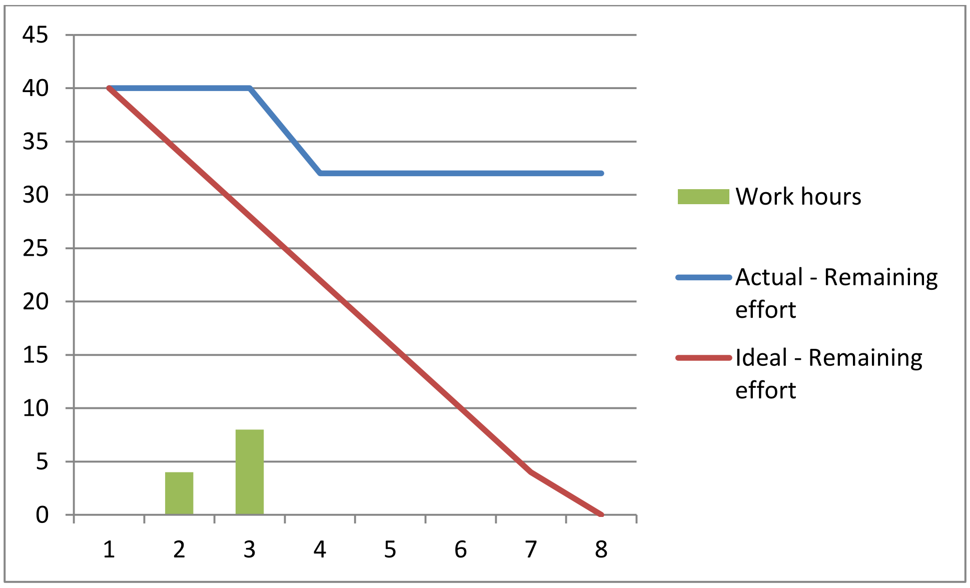


3.6 Sprint 3

25.02.13 - 04.03.13

Retrospective

We held a meeting about required backend changes, but little else was done in this sprint as other courses were demanding. We realized that a one-week spring is not ideal for the project, especially with the load from other courses sometimes consuming the entire sprint. We thus decided to change to two-week sprints. We requested access cards again so we could continue our work on weekends when situations like these come.

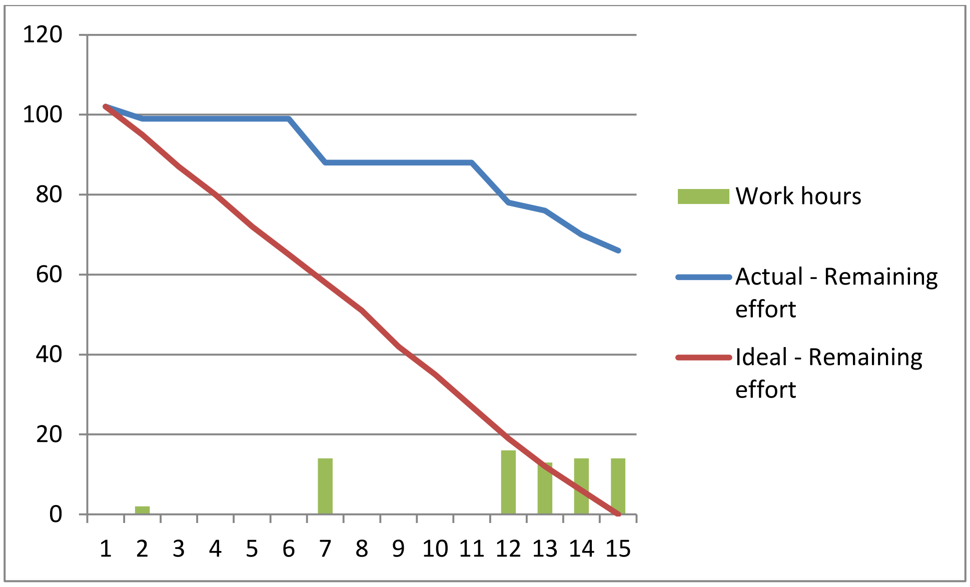


3.7 Sprint 4

04.03.13 - 18.03.13

Retrospective

Progress was impaired due to time-consuming assignments in other courses. We have allotted more time to the project on weekends to make up for the recent delays.

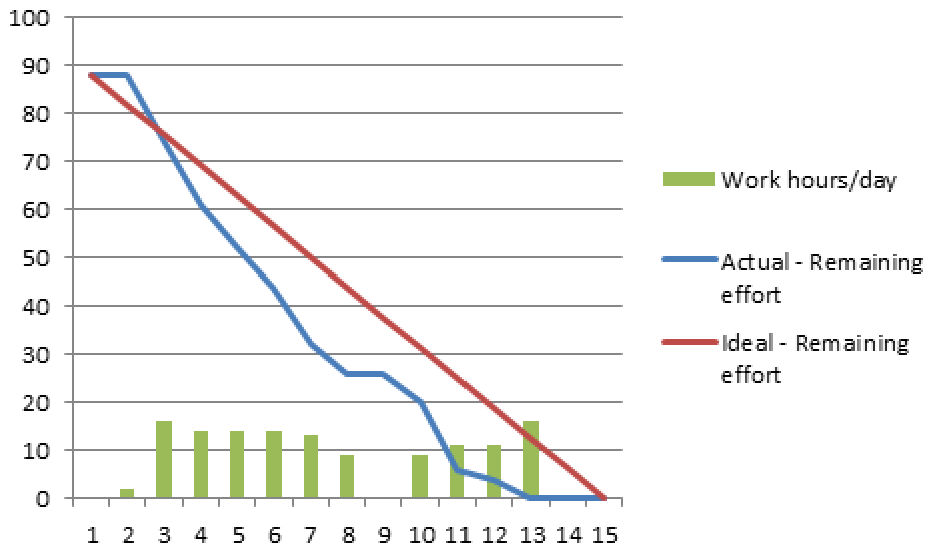


3.8 Sprint 5

18.03.13 - 01.04.13

Retrospective

The sprint went well, we now have a good feeling about how many story points we can handle.



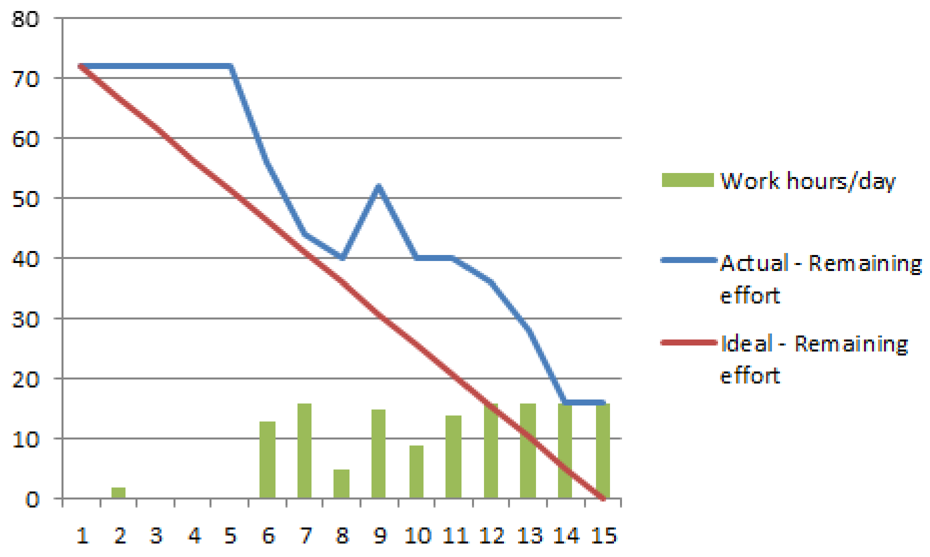
3.9 Sprint 6

01.04.13 - 15.04.13

Retrospective

We had trouble implementing specific features with the frameworks we were using and decided to change from jQuery to Google's SPA framework called AngularJS.

We had to cancel one story in the sprint to be able to implement the wanted features and change everything that was written in jQuery to AngularJS.

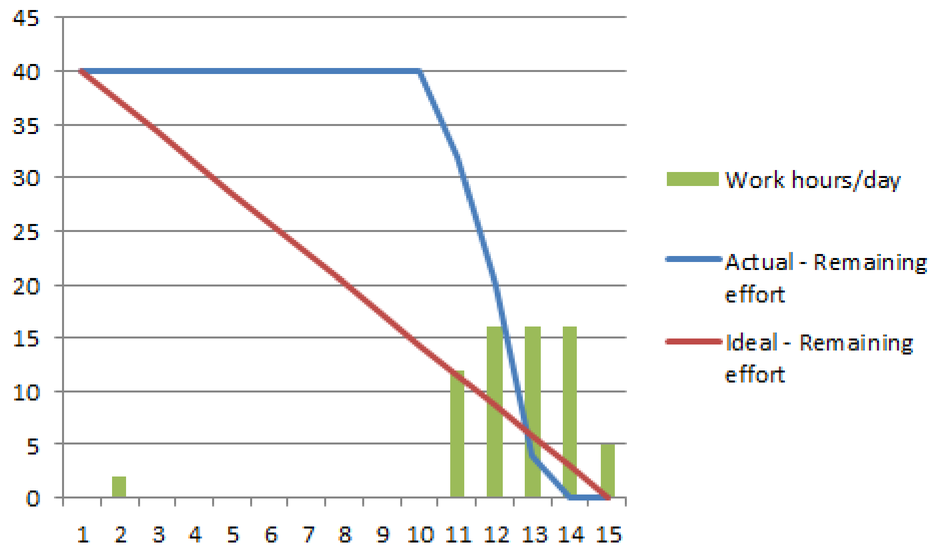


3.10 Sprint 7

15.04.13 - 29.04.13

Retrospective

The sprint took place at the same time as our finals and we only assigned tasks worth half our estimated velocity. Our estimate was accurate and we finished everything we set forth to do.

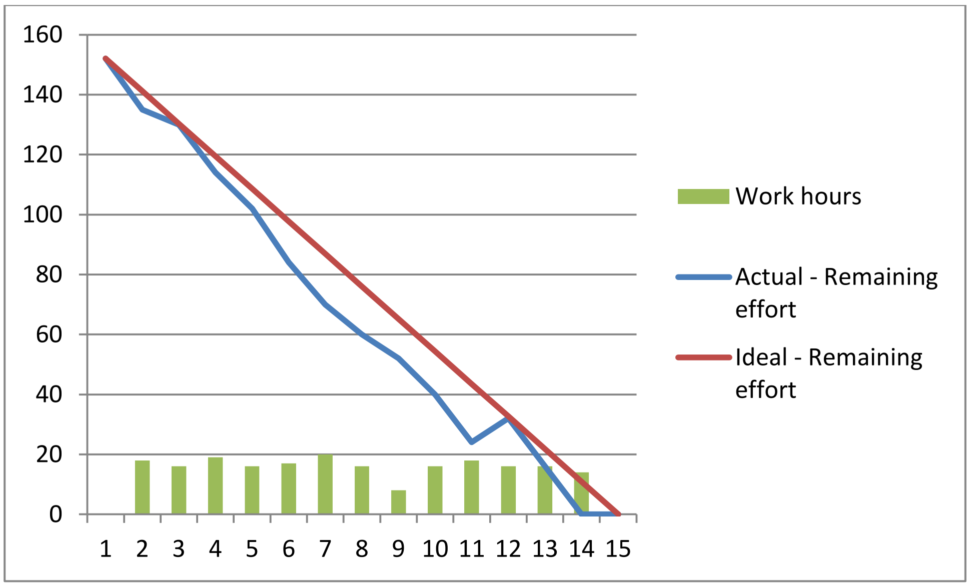


3.11 Sprint 8

29.04.13 - 13.05.13

Retrospective

The sprint went pretty well, the only hickup was that we implemented the post form incorrectly and didn‘t figure it out until after a test on it was conducted. Testing after someone closes a story was pretty beneficial in this case.



4. Conclusion

The main goal of the project was to facilitate the use of CREST and making it more accessible, thus improving testing capabilities and making development easier. This goal was reached, and the product owner, as well as other CCP employees to whom the product was shown, were pleased with the product.

Working on this project has been a very valuable learning experience. We tried pair programming and that became very helpful for us to understand problems faster and share our knowledge on what we had learned from the CCP developers on CCP‘s systems. At first we had a hard time adjusting scrum to our team effectively but were able to adjust to it as the project went on and found it very useful for managing the project workload. We also learned a great deal from working with the complex system that is EVE and it was very informative to see how such a big project is managed.

We found it rather difficult to familiarize ourselves with the OAuth autentication service, which seemed only to grow in complexity as the project progressed. At one point when we thought we had completed that part of the project, we realized that we needed a different kind of token from it, which changed the entire flow of communication. On the other hand, we learned a lot from these difficulties.

We were reassured as to the usefulness of the product by the reactions of CCP employees when presented with it. Judging from their reactions we believe it will come to serve its intended purpose, and be used by both CCP employees working with CREST as well as third-party developers.

Quotes from CCP employees:
‘I love this’ - Nicolas Tittley (Product Owner)
‘When do we get this?’ - Simon König (Quality Assurance)