Reference Counting

Project report
Project 1

Imperative and Object-Oriented Programming Methodology (1DL221), 2019/2020

Team 11: Adam Axelsson, Vlad Bertilsson, Carl Hasselberg, Ellinor Hallmén,
Ali Hoseini, Erik Junghahn, Miranda Kiss-Deáki

January, 2020

Table of Contents

1. Group name

Team 11

1.1 Participant list

Name	Email	Active dates
Adam Axelsson	adam.axelsson.4529@student.uu.se	191206-200117
Vlad Bertilsson	vladislav.bertilsson.1364@student.uu.se	191206-200117
Carl Hasselberg	carl.hasselberg.6020@student.uu.se	191206-200117
Ellinor Hallmén	ellinor.hallmen@gmail.com	191206 -200117
Ali Hoseini	ho.ali1010@gmail.com	191206 -200117
Erik Junghahn	Erik.junghahn.8642@student.uu.se	191206 - 200117
Miranda Kiss-Deáki	miranda.kiss-deaki.3122@student.uu.se	191206 -200117

Figure 1: Participant list

2. Quantification

Project start date	191206
Project end date	200117
Number of sprints, their start and end dates	Sprint 1: 191206-191223 Sprint 2: 191226-190110 Sprint 3: 190110-190115
Total number of new lines of C code written excluding tests and preexisting code	304 Lines in refmem.c 24 Lines in refmem.h
Total number of lines of test code	483 Lines in test.c
Total number of lines of "script code" (e.g., make files, Python scripts for generating test data, etc.)	71 Lines
Total number of hours worked by the team	About 420 hours
Total number of git commits	159
Total number of pull requests	14
Total number of GitHub issues	4

Figure 2: Quantification

3. Process

3.1 Inception

As work process we choose our own methodology with some elements of Kanban. The parts we choose from Kanban is the structure and visualization of work and shared leadership. The work is visualized with a Trello board with different columns like progress, to-do’s and completed tasks. That gives a good overview and members can easily see what needs to be done and pick a task from the board. In addition to the board we also used a burndown chart to be able to see our velocity.

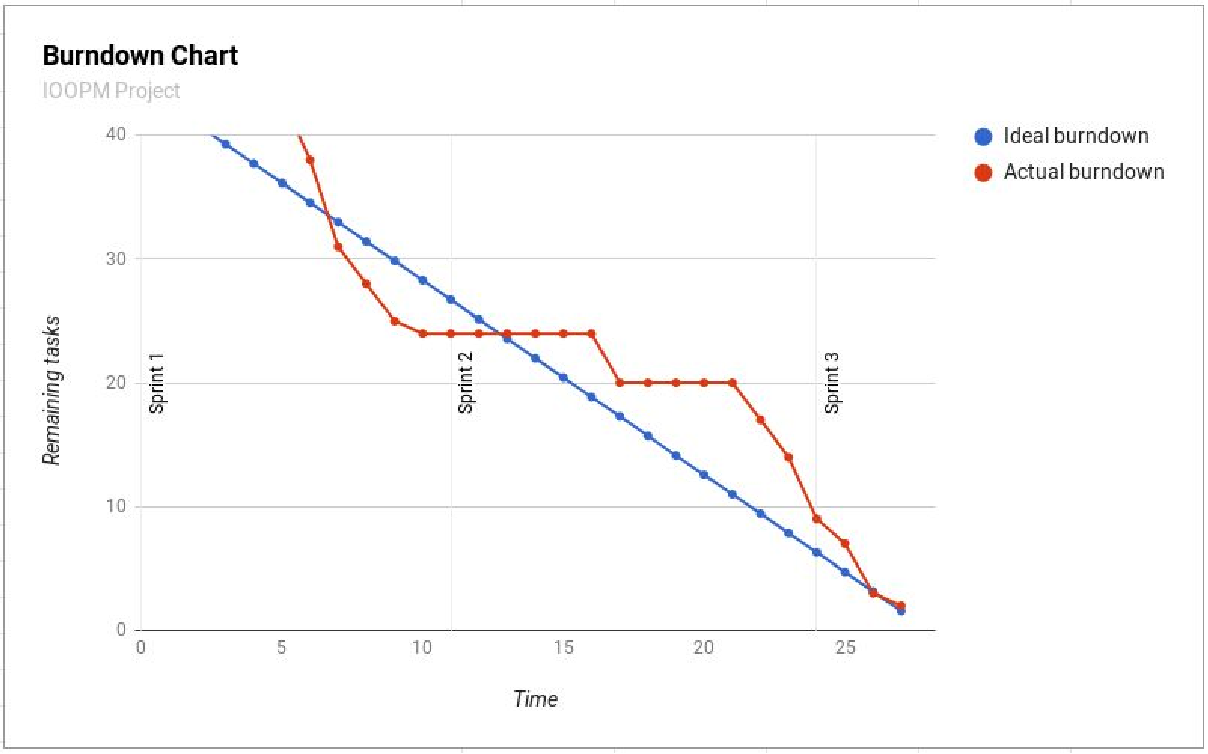
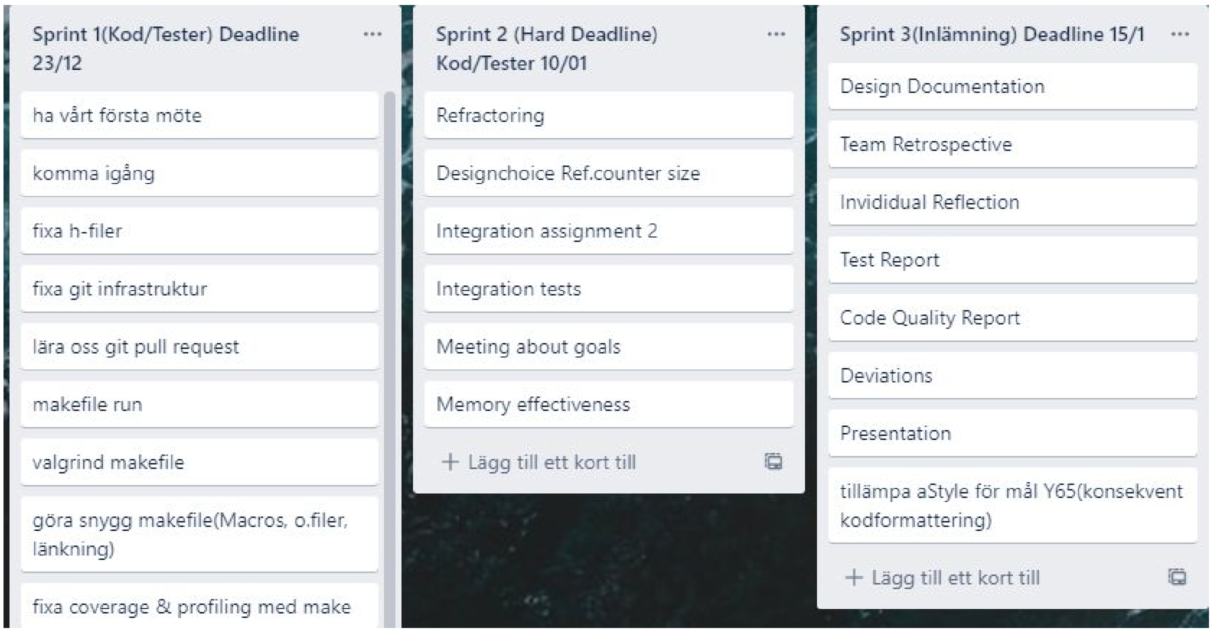


Figure 3: Burndown chart

At our second planning meeting we started to discuss what process we wanted to use. Since we were all acquainted with the process we learnt in this course we figured we would go for a mixture of that, a little bit of Kanban and our own ideas. That became our own process recipe.
We did not assign roles in the team, according to Kanban leadership can be shared among members in the team so we kept the leadership role fluid in the group.

Since pair programming has been a big part of this course, which we have found positive, we decided to divide the group into three teams. But we all agreed that we in the early stage of the development should work all together in school. Classroom 2244 became our “office” and “projector programming” became a big part of our work methodology. With one person sitting in front of the laptop, which is connected to the projector, and the rest watching we found a lot of ideas surfacing since everyone can see what is going on. The ideas flow freely in the group as the person sitting in front of the laptop records the ideas into the code. The person coding is alternated and everyone will sometimes code all at the same time (on their own laptops) when there is a bug or memory leak we want to try different hunting strategies for. The whiteboards in the classroom are used for spicing discussions, design choices and implementations.

After a couple of “projector programming” sessions with positive outcome we felt that it was a very good way to work so we decided to continue in that track. The three teams we divided the group in earlier was almost never used, but we have been working individually as well with smaller tasks, sometimes at home, sometimes in the classroom.


Sprint 1(Kod/Tester) Deadline 23/12
- ha vårt första möte
- komma igång
- fixa h-filer
- fixa git infrastruktur
- lära oss git pull request
- makefile run
- valgrind makefile
- göra snygg makefile(Macros, o.filer, länkning)
- fixa coverage & profiling med make

Sprint 2 (Hard Deadline) Kod/Tester 10/01
- Refractoring
- Designchoice Ref.counter size
- Integration assignment 2
- Integration tests
- Meeting about goals
- Memory effectiveness
- + Lägg till ett kort till

Sprint 3(Inlämning) Deadline 15/1
- Design Documentation
- Team Retrospective
- Invididual Reflection
- Test Report
- Code Quality Report
- Deviations
- Presentation
- tillämpa aStyle för mål Y65(konsekvent kodformattering)
- + Lägg till ett kort till

Figure 4: Trello board with sprints

After some discussion how the sprints were going to be divided we figured that the project work could be split into three big parts leading into three sprints. One sprint for the coding, one sprint for refactoring, one sprint for reports/presentation. We created three columns on the trello board, one for each sprint.

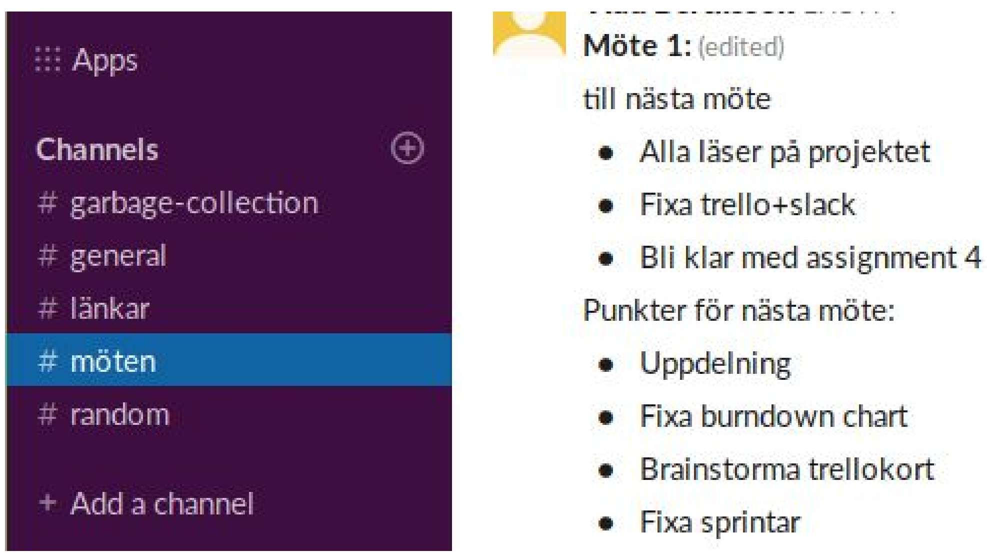
Our next goal was to set dates for when each of these sprints needed to be done. Factors that had to be considered was:

- The amount of time we thought each sprint would take,
- How early relative to the seminars we wanted the entire project to be done
- Winter holidays
- Upcoming exams for every group member

We then ended up with a deadline date for each sprint as seen in figure 4. We wanted to set these deadline dates early as they pushed us to finish each sprint and in that way balance the workload. If we didn't set these deadline dates early, it would be hard knowing how much work would be needed for each sprint which could lead into postponing of the work. We took inspiration from the structure of how the assignments of IOOPM were. We felt having a “soft deadline” and “hard deadline” made the stressfactor lower as compared to having one deadline. Therefore our idea was that sprint 1 was going to be our “soft deadline” and sprint 2 was going to be our “hard deadline” for when the code needed to be done. This lead to that sprint 2 contained lesser cards and sprint 1 contained more since all non-completed cards from sprint 1 would later be moved to sprint 2.

Another tool that was important in our process was Slack. We used it to communicate and for storing information in a sorted way. Some of the channels we used for this was:

- Links - storing links to trello, burndown chart, reports etc
- Meetings - storing meeting protocols
- General - for work-based matters
- Random - random things and other announcements (sickness etc)



Möte 1: (edited)

till nästa möte

- Alla läser på projektet
- Fixa trello+slack
- Bli klar med assignment 4

Punkter för nästa möte:

- Uppdelning
- Fixa burndown chart
- Brainstorma trellokort
- Fixa sprintar

Figure 5: Slack

Since the work was very structurized we managed to follow our chosen process quite well.

3.2 Implementation

Key plans were made during meetings but new decisions could be made during the working sessions if someone came up with a good idea, in that case we would record it in one of the Slack channels. The chosen process however did not change much, except for the small groups we did not use.

Strengths and successes
We ended up implementing a quite good work management system. The key to that was keeping work structured and in that way keeping everyone in the team on the same page. Another important aspect was having good communication in the team. Since we worked all together in a classroom questions could easily be answered and solved directly. During the christmas break when not everyone was in town, Slack was the main channel where communication took place.

Weaknesses and improvements
It might have been more effective to work in small groups instead of a big group. If that actually would have been an improvement is hard to predict since we did not try it. Although, if we were to do the exact same project again, then we would probably work in small groups to speed up efficiency, since we are now familiar with the product.

We did not have a protocol system of what we exactly have been doing that day so if you are not there you might miss out on something. We did use github of course but that does not explain on a detailed level what choices has been made and why. It was however managed quite good by orally explaining the things that were changed and communicating on slack but it could be improved, since it is easy to miss something. Especially during the holidays when not everyone was in town this was a tricky part, since you might have to remember to explain what has been done the last few days and what decisions have been made.

The projector programming could be improved by having a wireless keyboard connected to the computer that is projected onto the screen. The keyboard could easily be sent around in the group if someone gets an idea.

4. Use of tools

The tools we used were very useful for our project and team working. It helped us to organize, communicate and debug. If we were to start over tomorrow we would use the same tools because all of them were necessary for our project. But the more tools the merrier, so in addition to the tools we used it would have been nice to have a time tracking tool for the tasks. There is probably an extension tool for that in Trello. A more advanced Kanban board would also be interesting to try out.

The tools we used in the project and the usage of each tool is stated below.

Make: Automation tool used to compile executables.

Valgrind: Tool used to detect memory leaks in the program.

Trello: Used to create a board of the work divided into smaller tasks, to keep work structurized and to set up goals (sprints).

Google spreadsheets (burndown chart): Used to create a burndown chart to keep track of our progress and velocity.

Slack: Communication tool that we used to keep in contact with team members. Also for structuring work in different channels.

gcov: Code coverage tool to test code coverage on our program. This tool help us to know percentage of test cases covered.

gprof: We used gprof which is a performance analysis tool to profile our program. With gprof we could see where the most of the time is spent during the program execution.

gdb: This tool uses to debug code. We had some bugs and segmentation fault during our coding process and therefore we used gbd to find where we had bugs.

5. Communication, Cooperation and Coordination

One of the first things we did was setting up a slack group where we had channels for planning meetings, discussing code and sharing links. In the beginning we discussed how people were doing on the assignments and tried setting up a schedule for when to start with the different steps of the project. Later we used our Slack chat for discussing different solutions and approaches.

Between meetings and coding sessions we discussed smart ways to approach the problems we faced with our friends outside of the group, both sharing and taking advice on the various tasks.

We had six meetings with our “Team coach” Maria Andreina during the project. We discussed our progress and asked for help on different problems we encountered. It proved to be quite hard to ask for help sometimes, since if we didn’t understand something it wasn’t easy to explain what we didn’t understand.

Most of us went on break a few days prior to christmas which resulted in some silence in the group for about a week around christmas but some of us discussed and decided to come back and do some more work on the 26th until new years.

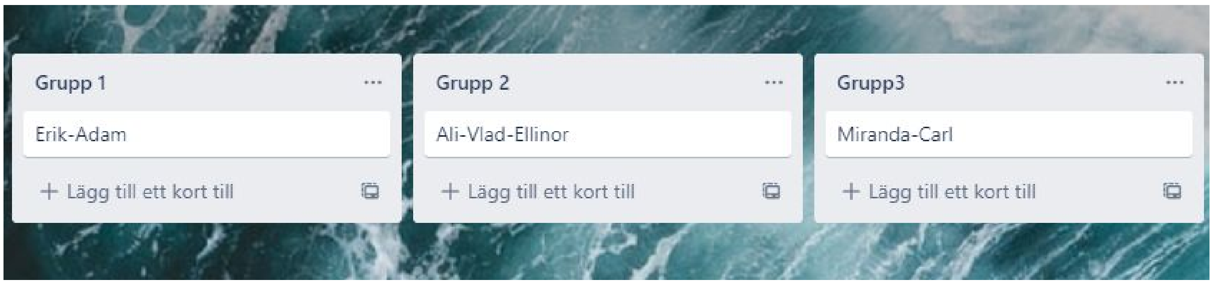
The majority of the team members had a lot of work to do beyond the project. Many were not done with Assignment 4, and most of us had exams around week 2 which created stress in the team. To solve this stress issue, we decided to put the project on pause for a week after new years to let everyone finish up their work that did not regard the project. It was a good decision because it definitely relieved some stress, and also meant we could fully focus on the project during the final week before the deadline.

6. Work Breakdown Structure

On the first meeting together we discussed on how we wanted to work with the project and move on. We ended the meeting knowing we were gonna use trello and use cards that were gonna be distributed across smaller groups inside of the team. Our idea was that these trello-cards should contain smaller tasks which we could complete one by one. Using this method would give us a sense of accomplishment each time we completed a smaller task. And if the cards were load-balanced it would lead to a clear understanding of our progress of the project.

We also had an idea of that each of these trello cards could be used as a checkmark for the mandatory burndown chart. This burndown chart would later on help us to visually see our progress and could be used as a motivation tool to push us to work more.

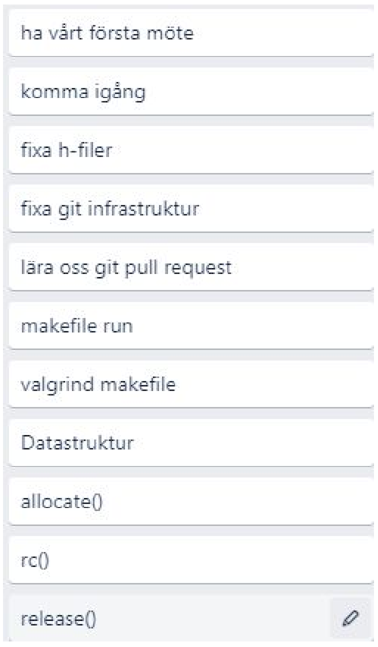
At meeting one, nobody in the team had read the project description which lead to that it was impossible to come up with content to the trello cards. After meeting one was done, we said that every group member should read the entire project description and be ready to brainstorm cards during the next meeting. We thought that having a few somewhat reasonable trello cards early, would make us feel more prepared of the workload in the time we had ahead of us. We also said that during meeting two, we should divide ourselves into smaller groups of 2-3 people. Our idea was that each group could pick a task from the trello-board and have full responsibility to make sure that task got completed. Our reasoning behind this method was that nobody would feel that they worked more than others and that the workload was balanced evenly for all team members. This is good to avoid internal team conflicts regarding time spent on the project. It would also give each group its own responsibility which leads to efficiency.


Grupp 1	Grupp 2	Grupp3
Erik-Adam	Ali-Vlad-Ellinor	Miranda-Carl

+ Lägg till ett kort till
+ Lägg till ett kort till
+ Lägg till ett kort till

Figure 6: Group Cards Trello

At meeting two we sat down to divide and distribute the upcoming work. Our idea was to simply pick any task that we knew that had to be done at some point, and create a card about that task on trello.


ha vårt första möte

komma igång

fixa h-filer

fixa git infrastruktur

lära oss git pull request

makefile run

valgrind makefile

Datastruktur

allocate()

rc()

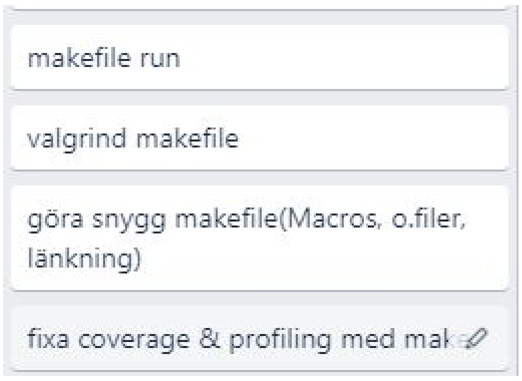
release()

Figure 7: Early tasks on trello

Since we knew that the program had 8 functions that needed to be added from the given .h file, we simply added each function as its own trello card. We then added things we knew that had to be done as trello cards, such as (Creating a makefile, creating a make test, creating data structures, creating a make valgrind, creating a git infrastructure, etc). We then added trello cards for each handin given on the project overview page. After the brainstorming was done, we had roughly 20 trello cards.

Our next task was to divide the current trello cards into smaller tasks. Our reasoning was that if you have smaller tasks, it is easier to start that task since you know each small task take a lesser amount of time than a big one. Having a small task, would be more motivating to start working on compared to having a large task. This would lead into that we would never have any back thoughts of not starting a task simply because of the size of that task.

During the project we changed and added cards a lot, for example in case a card was too big we could divide the card into two smaller tasks. And if we managed to do some subwork on a bigger card, we created a card for that subwork. One example of this was our makefile tasks:


makefile run

valgrind makefile

göra snygg makefile(Macros, o.filer, länkning)

fixa coverage & profiling med mak

Figure 8: Makefile tasks

Each makefile label could be its own task, and since our goal was to divide tasks as much as possible, we did one trello card for some of the labels in the makefile.

When this structure was done we decided to start working on the project together as a team. We did this since there was no way to split up the workflow since every function given by the .h file that had to be written was dependant on having a data structure. As explained in the Process section group, working all together was very positive, so we continued with that throughout the whole project.

7. Quality Assurance

Considering the fact that each memory allocation and free aside from “strdup’s” are done in the “refmem” file and not in the initial files were we use the memory, we believe we have implemented the correct thing. We are always able to allocate new data structures and assign them their own reference counter which works as intended and frees the structure when the last reference to it was released.

Vladimir began implementing some unittests once we got the example.c file provided to us to work without leaks. We wrote all the tests for the reference counter before integrating it into assignment 2. Before we started with the integration we also made sure to run the tests for assignment 2 without our reference counter to see how much data was allocated. Once the integration was done and we had made sure that our previous tests worked for the new implementation for the assignment we compared the variety in data allocation. We realised that the amount of data allocated with the reference counter implementation was reasonable.

We only used pull requests a couple of times as an experiment but ended up not using it anymore. Since we ended up all working together each time we programmed everybody checked what was being updated to each branch and there was no need for additional reviews from the team-members. We figured 14 eyes is as secure as one pull request.

8. Reflection

We consider our biggest win to be how we managed to get together and work together as a whole team. Projector programming was especially good, and fun as well. We had no problems with keeping a good working schedule and getting a sufficient amount of work done every week. Probably thanks to the time we took in the beginning of the project to structurize the work.

Satisfaction regarding process: 6
Our biggest fail was not reaching our scheduled deadlines. Therefore the rate of satisfaction stops at 6. For example we decided we were supposed to be done with the “refmem” file the day before christmas. This however ended up taking a little bit longer than expected and the “refmem” file was more or less done by the 27th. The same thing happened with the integration and testing which were supposed to be finished by 10/01 but took until 13/01. Overall we missed our own deadlines by a few days but we kept a continuous working schedule for the most part. The last sprint for writing reports could have been longer, it took more time than expected.

Satisfaction regarding delivered product: 5
We consider our delivered product to be very satisfactory to the given requirements. The program works as expected and the amount of data allocated with the reference counter doesn’t exceed twice of what was originally allocated by the assignment it was integrated with for the most part. In the worst case however when we only allocate 8 bytes we are still going to get an overhead of at least 32 bytes which goes against the specification.

Satisfaction regarding quality assurance: 5
In order to assure the quality of our delivered product we wrote a heap of new tests using the cell structure given to us as an example as well as verifying that the tests from the second assignment still worked. The result of us then checking for line coverage, memory allocation and time complexity of these tests give us a fair bit of confidence in that the quality of the program is assured. However we could always test more cases in order to be 100% sure of our codes quality.

Team retrospective

As stated previously we only did a few pull request in order to test the functionality of it. However as we did not end up working with pair-programming and always wrote our code together in a large group we decided that 14 examining eyes is better than a two eyed pull request review. Before every merge into master we discussed and reviewed the changes.

Time distribution
Unfortunately we don’t have an exact time log for every group member, but since we wrote up the dates when we worked with the project and worked on average of 5 hours each session we can approximate the time. 420 hours work in total for the whole group, so if a company would hire us, it would only cost them 84 000 kr for this program. Supposing every person has a salary of 35 000 kr.

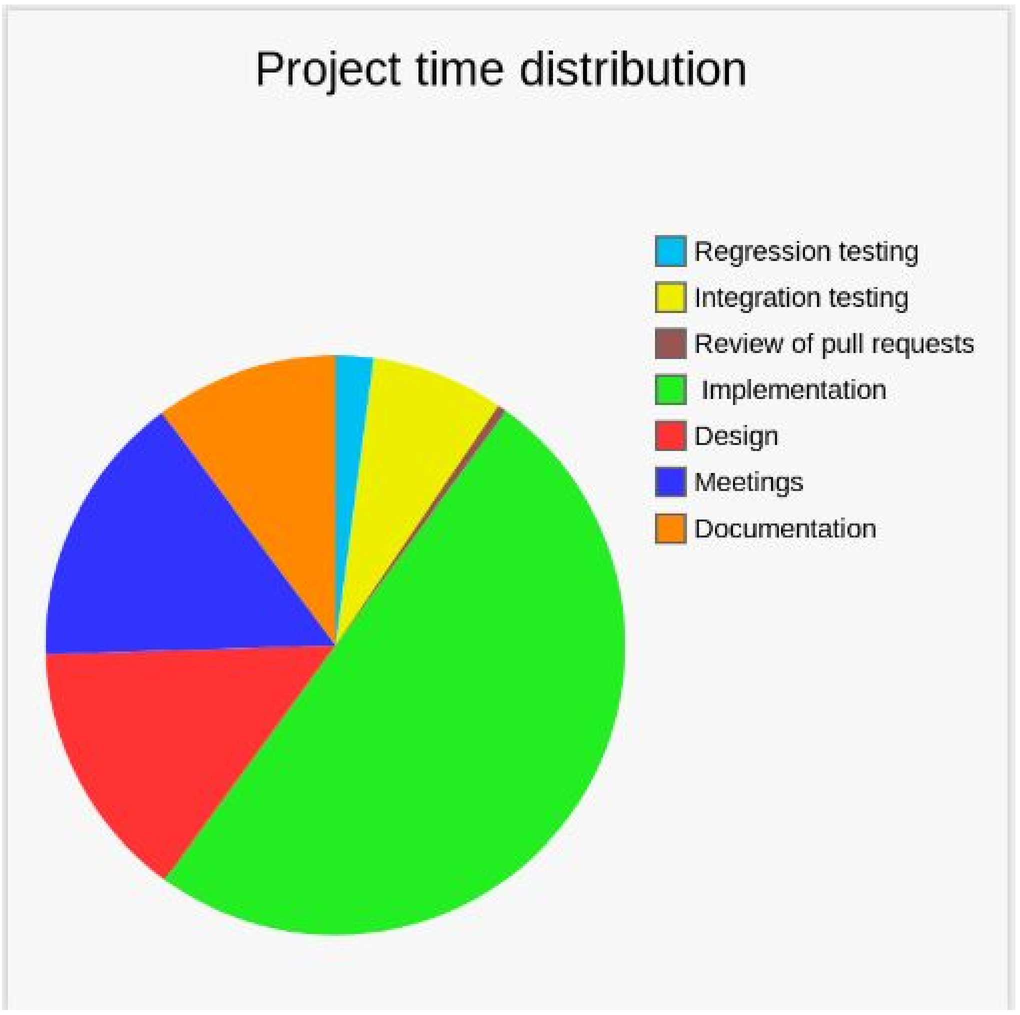


Figure 9: Pie chart over time distribution

The time distribution of the project is stated in figure 9. Implementation of the program took most of the time, and pull requests least of the time since we only tested it a few times in the beginning.