Project Web Services

Matthew Anthony Peterson
Ivo Vasilev Nekov
Bogdan-Iulian Vasile
David Bertus-Barcza

Table of Content

How and why did you choose your SDM?

Before starting to work on the project we organized a meeting to choose which System Development Methodology do we want to work with. We agreed that each of us should come with ideas of different methodologies which we prefer the most. At the meeting after some negotiations we decided to work with the “AGILE” methodology.

We prefer “AGILE” because it breaks PBIs into small tasks which minimize the planning time frames. The items are made into sprints. Each sprints involves all the team members to work in all functions (planning, analysis, design, coding, unit testing and acceptance testing). At the end of the sprint a working product is demonstrated to stakeholders which minimizes overall risk and allows the product to adapt changes quickly.

We like “AGILE” because it is focusing more on direct verbal communication rather than writing to each other. The team members are usually working at the same area which is ideal for us. Also there are small teams where the team spirit can be developed easier than in big groups.

We are going to use Trello board to set up the tasks and prioritise them.
We can use some specific tools and techniques such as pair programming, design patterns, code refactoring, automated unit testing to improve quality and enhance product development agility.

From least year’s studies we have experience working with “AGILE” which makes planning and working easier.

Did you make any changes in your SDM? Which and why?

The only change we made is that we are not going to meet with our product owner after every sprint because the expectations are clear and we made decisions on whether or not to create many of the diagrams that are outlined in UML, the reasoning behind these decisions will be shown below.

Describe and show how you worked with each of the focus areas

Start Up

Our client is Jannick Toftegaard Jensen from Izinga. Our solution’s purpose is to collect and structure log files (any type of log files) from multiple vendor’s systems for displaying significant entries in one system to provide a better overview for actions in multiple systems.

In detail, our software has to give an overview of plenty of systems from their log files. Our system should have filters which helps the user to structure the data from the log files.

Our other requirement is, while the log runs in realtime and data is being written we want to be able to call the service every 5 seconds and for it to read the new lines and send an output to the system if some specific entries occurs.

High-Level Design

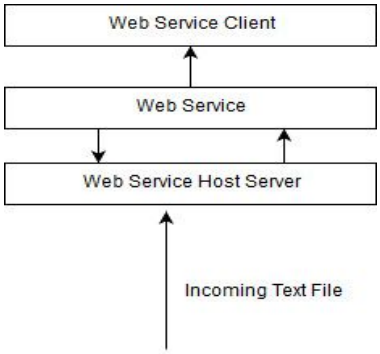
In the high-level design we defined how our major components will fit together and interact with each other when the finished pieces are created.

The requirements regarding the enviroment in which the app will run were specified as Windows operating system and hardware specification were not specifically defined because the application won't be that resource heavy. There won't be any database present in the current system however such an extension won't be hard to implement

as the app is developed using the SOLID principles.

We are using the service-oriented architecture. We have a client and a service that is running on a local server. The server is fed with text files playing the role as log files coming into the system and onwards the data is being processed and the new inputs are entered in an already existing Log File.

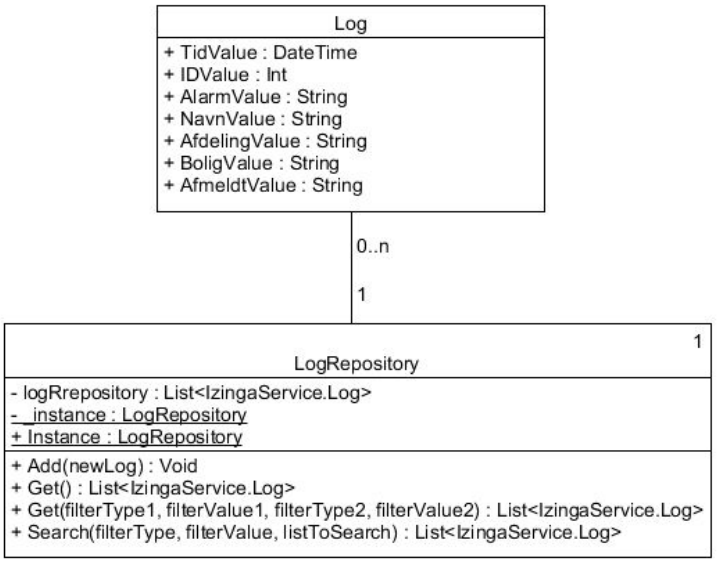
During the high-level design we managed to split the tasks for the different components between the team members. However if a task was difficult for any of the members it was discussed and help was provided.



Low-Level Design

In the low-level design phase we did not feel the need to create SDs because the solution we were making was not going to have any complex functions, there are not many methods or classes that will be communicating. We basically just have two methods that do all of the work in the solution.

We did, however, create some DCDs for the classes we did have. They are both very small and simple, but these were the only classes we had in our project. The Log class is a class created through a DataContract in the Service, and then we have the repository to store them in memory. We drew this diagram to better visualize the classes that we would need and to allow all members of the group to know what classes we have and what they will do.



Development and Testing

We have made use of several tools during the development process such as Google Drive for sharing documents and storing the artifacts or Visual Studio combined with C# for a proper development environment.

The hardware we have used was our personal machines.

Source control played an important role during the process because we had different members working on different parts of the system, therefore, GitHub was our ally.

As source code formatter, Visual Studio does a great job regarding it so no additional formatter required.

Our training was based on internet materials such as videos, blogs and tutorials.

The algorithms used in our program are designed to be simple, effective and easy to understand taking into account that our main purpose was to exercise and polish our skills. We tried to make our code self-documented using descriptive names for classes and methods and to use comments when it was necessary.

The process itself was relatively slow because we did not have a lot of experience with the platform, so this required a lot of individual research and remodeling some of the logic. Using the SCRUM board we divided the tasks between us, a fact that allowed each of us to be involved in the process.

Taking into consideration the reduced complexity and size of the program, all the testing was made in the console using debug mode.

Deployment and Maintenance

The project in itself was an exercise, giving the small amount of time we had to complete it and the lack of information.
Therefore, we do not have any deployment or maintenance strategy set up for it.

Describe your quality criteria

We used criteria to determine if the diagrams we created were done correctly and that we all followed the same conventions for creating the diagrams. We wanted to create the criteria in writing so there would be no confusion between group members on how something should be created.

Below is an example of our criteria for the DCD:

Criteria for DCD

Purpose:
Further your understanding of how the software should be developed and use it to communicate it to other developers. Helps to get an overview of the system and to see what design decisions could be made. Needs to be updated during the development phase as you expand on your system.

Criterias:

- The DCD contains only software classes and not the conceptual classes.
- Public methods or variables have a + sign
- Private methods or variables have a - sign
- Methods and Properties are seperated by a line
- Singletons have a small 1 in the top right corner
- The parameters and return types are shown for methods
- The type is shown for properties

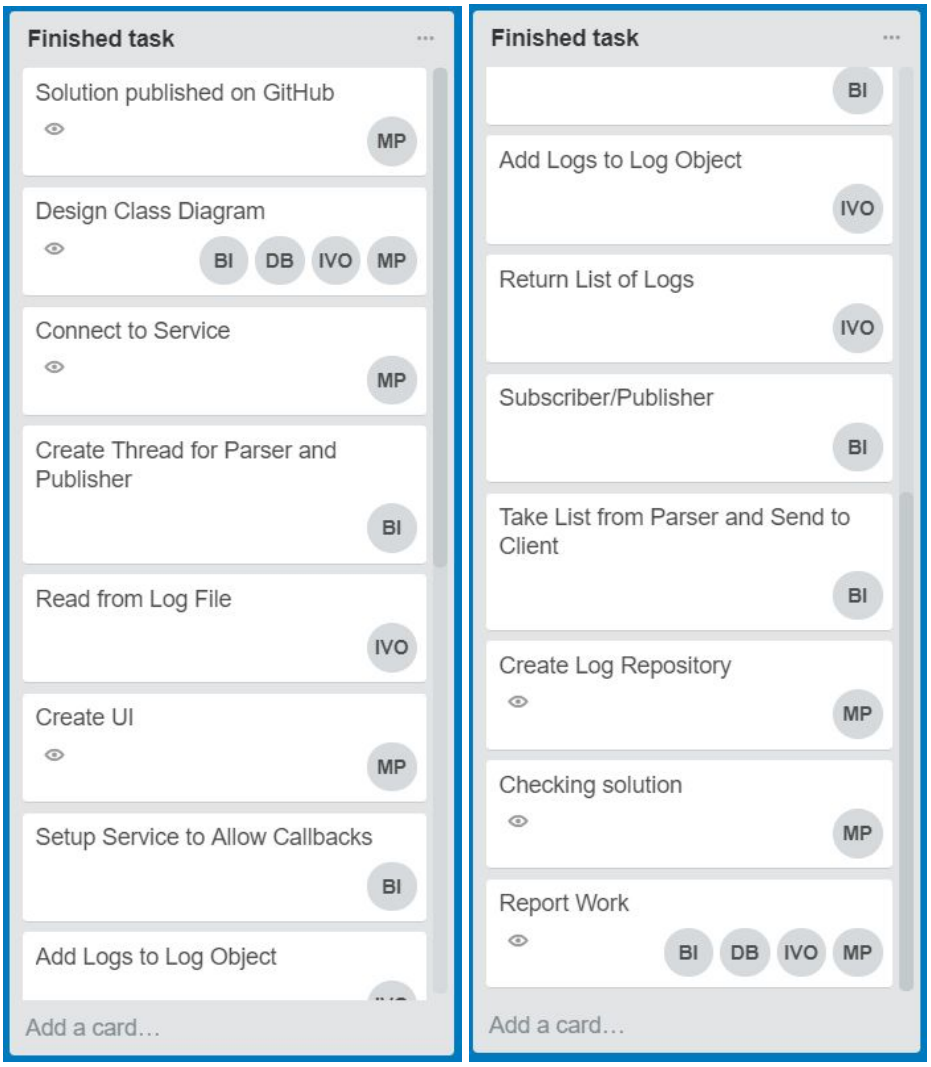
All the criteria for diagrams we made were in a similar format to this, just with changes that pertain to each respective diagram.

How did your SDM support traceability

We were able to see exactly who did what task and the diagrams we made allowed us to see how we progressed through the project. We used github for the project, and we made sure to commit as much as possible, so that it would be possible to follow the progression of

the code as we worked. We also wanted to make sure that we could go back to an older version if need be.

Below we have some of the tasks that we had in our scrum board:



Evaluate your choice of SDM

What went well?

We were able to assign tasks to each other. This way we were all able to know what the others were working on. There was also a feeling of accountability in the group because we all could see who was assigned what and if they had completed it.

What could be improved?

We weren't able to split up the PBI into the different sprints because the project was so small and it wasn't very long either, so we did not follow that exactly. We didn't create tasks for every single thing we did. We had a few diagrams that we worked on that didn't have a task for.

How would you make it better next time?

We would like to implement more parts from the other methodologies. We mainly focused on the parts that we had learned from last year, but we would like to, in the future, implement some other ways of working from the other methodologies because we believe they could be helpful in creating a high quality solution. We also want to create tasks for everything we do.

Technology

We chose to self-host our service our web service. The main reason for this is because we felt the most comfortable doing it this way. It also allows us to have more control of exactly how it will run, we weren't able to fully take advantage of this, however, because of time constraints. Self-hosting also runs faster than a cloud host because it is all done right on our own machine, we thought this was important because we were mainly focusing on learning, and we didn't want to be waiting around all the time whenever we wanted to test it.

The cons to this decision were that we don't have complete knowledge of all aspects of self-hosting, so we don't know all the things we can do with it, this is less a con of self-hosting and more an empty space in our knowledge. Self-hosting also requires that you have your own hardware and software to host it with, this wasn't too bad for us because we only had to host one small service on our computer, but it did mean that we couldn't have it running 24/7.

In terms of security challenges in our system, our system doesn't use a database, so we don't have to worry about any sort of SQL injection. It does, however, use a text file to store

the logs persistently. This can be very insecure if someone is able to access the log file and change its contents. We considered trying to make the file get created by the system as read-only, but we were unable to implement this feature. We were unable to check the inputs of the user using a whitelist/blacklist approach, but that is something we would have done if we had more time.

The system will throw errors for many different things. One example being that if we try to read the entire log file at once, it will throw a “The remote server returned an unexpected response: (413) Request Entity Too Large” error because we are trying to send 55,000 lists to the client. We were unable to fix this problem other than making the log file smaller.