Group03 Justification of Choice of architectural styles

Reasoning for not choosing Blackboard

The given task asked us to choose between the styles **Publisher/Subscriber**, **Pipe-and-Filter** and **Blackboard** and to implement two of them in combination**.** After weighting the given styles against each other, we decided on a combination of Publisher/Subscriber and Pipe-and-Filter.

The decision to exclude the Blackboard pattern, is based on a series of reasons, which will be shown in the following section. Looking at the Blackboard-pattern, we realised that it could suit the needs of the project, since it provides reusability and changeability, two important attributes for the given task. However, there was another important factor that we had to take in consideration. Even though we felt familiar with it on a theoretical basis thanks to the former course, we were lacking experience in working with it in the practical implementation. After we received examples for the use of this pattern, we investigated them thoroughly and concluded, that we would have to invest much more time and effort into understanding and implementing it correctly. Knowing about the importance of a good architecture, and the benefits of a mutual understanding of it, we decided against using Blackboard, to avoid success-threatening problems later in the development.

Reasoning for choosing Pipe and Filter

The pattern is suitable, when a process can be described as a set of independent steps. Since we want to apply filters on the data, which are independent of each other, this pattern seemed like it would offer valuable capabilities to solve the given task. This pattern also offers a high flexibility in adding and removing steps in the process, which can be very useful if you want to solve different problems with the same given data.

Reasoning for choosing Publisher/Subscriber

The Publisher/Subscriber pattern enables an application to publish messages to several independent receivers. Since we are working with a central data source (requests and data from Västtrafik), that broadcasts data to different entities that can visualize it, Pub/Sub seems like the most viable option. Our visualization entities are also running on different platforms and even in different programming languages. While this can be a problem for other styles, Publisher/Subscriber supports it very well, thanks to the low coupling to the subscribers.

The use of this pattern also allows to send information to different consumers, without handling their real-time responses. Since the use-case is not intending interaction between the clients and the publisher and one of the requirements was not to use the Client-Server pattern, we used this technical limitation/ability to our benefit.