# Software Patterns:
# Review of second assignment (Version 0.3) of Group 2

Thomas Hoeksema, Maarten Kollenstart, Toon Albers,
Michel Medema & Jasper Mohlmann

January 2016

## General notes

We will first provide you with some general notes before going into detail:

- Please mention a scope somewhere (i.e. "Docker Engine"). Which parts of Docker are included/excluded, and why?
- Try to proofread each other's texts to pick out any errors. Not only content-wise, but also spelling and grammar wise, because there are quite a few of these mistakes in the document. In some ocassions, there is also duplicate text or simply duplicate information being explained twice in slighly different ways.
- The amount of white space in your document is distracting and too obvious, try collapsing some content onto the same pages.
- You refer to the GitHub repository all the time (bibliography item 10). Please give a more detailed explanation of it somewhere, otherwise you can also just refer to docker.com for all the documentation and other information. If we go to github.com/docker/docker it doesn't show directly the information you describe.
- Include a glossary of terms that the reader may not understand immediately from the surrounding text (related to Docker mostly).
- We noticed many small improvements during the presentation. Keep it up :)

We will now give you a summary of the most important review points per section of your document. Please refer to the other attached document with comments for a complete, specific list of in-line comments in your document.

## Introduction

- Docker can run on Windows and Mac OSX, although through a VM using Docker Machine. Docker Engine itself can only run on Linux. Please make this difference (more) clear.
- You miss a short description of section 4 when explaining the structure of the document.
- Bullet points could make some paragraphs easier to read.

## System Context

- Both section 2 and section 2.1 are titled "System Context". We suggest renaming the second one.
- You start with an introduction to Docker, then explaining operating-system level virtualization, and then you kind of introduce Docker again. It seems like you forgot you already mentioned Docker. A suggestion is to start with either virtualization and then Docker or Docker and how it is different from the full virtualization.
- Is the part about the build process and its layers appropriate here? Seems more suited for the software architecture to us.
- The order and structure of information presented in this section could be improved to make the section more readable.
- Maybe add some information about the Docker company.

## Stakeholders and Concerns

- Are cloud providers and the Open Container initiative not external stakeholders, that have a far less impact on Docker than other stakeholders? Why don't you use a SH/QA matrix with points for stakeholders to divide on the quality attributes?
- It may be necessary to come up with some better way to derive the key drivers, other than just which QA has the most connected SHs. Some SH seem more important than others. In fact, if you implement our suggestions regarding adding performance for software developers and removing portability for the Open Container Initiative, most QAs will have a degree of two or three.
- The part about performance in the portability key driver is out of place. The fact that containers are small, yet have high performance, doesn't inherently make them portable. Other than that, wouldn't portability most likely have a negative impact on the performance, as the code can't be optimised for specific environments?
- Make sure that you only cover the concern itself in the explanation of the concerns, try not to already include how Docker solved this in their architecture. Also, make sure not to derive concerns from the architecture (which is the inverse of what should be done).
- Reliability for Software Maintainers seems to be referring to talking about the reliability of Dockerized applications, but the Cloud Provider concerns appears to be more about Docker (Engine/Machine/Swarm) itself.
- We think that another reason why Software Maintainers care about security is the fact that user applications should run separately without having access to each other's resources and data.
- For Cloud Providers: don't they want performance, and what is the clear difference here between portability and compatibility? (latter also holds for The Open Container Initiative)

- "Docker Hub" is introduced first in the Portability key driver without further explanation on what it is. We know what it is, but the average reader won't.

## Software Architecture

- You mention you will present three views. Firstly, only two views are present. Secondly, the assignment on Nestor specifies a maximum of two. Ask your coach for their opinion.
- Try not to repeat information from earlier sections, such as mentioning details of the system context here again.
- The Docker daemon doesn't really start unless you tell it to. You could add it as a start-up service, sure. But essentially it's just a `docker daemon` command.
- We assume you will expand your process view with more processes.
- It is not clear that Figure 4.1 and 4.2 relate to each other, apart from the fact that it is mentioned very briefly in text.
- Figure 4.3 can be made easier to read (see comment).

## Pattern Documentation

- Overall, this section is in need of some review and refining, and does not fully map onto what you presented during the presentation. (Which was arguably better at some points.)
- Perhaps give a short introduction of the patterns on what they do and how they work. At least not jumping into the scheme. A image of how the pattern comes forward in Docker would also increase the clarity of the patterns, since for the most part the existence, traceability and source of the patterns are not very clear at all.
- You don't use the same pattern documentation for all the patterns. In some, the Implications part is missing, and in others the Solution. And try to stick with the intended goal of each part of this documenting structure, for example in the rationale you now state almost in each pattern how the pattern is implemented in Docker but not what the reason can be for Docker to have implemented the pattern that way.
- For the Client-Server pattern, isn't security also negatively affected by this pattern, as you open up an entry point to the daemon? In the presentation, you mentioned an increase in security by using this pattern but that was mainly because other patterns that are working together with the client-server pattern increase the security. It makes it easier to evaluate the patterns if you stick with the implications of only the described pattern and not the related patterns.
- For the Layers pattern, you describe the layered file-system as a layers pattern, but the layered file-system is more a data structure than software subsystems. In your presentation you have mentioned other layers that seems to be suitable for the layers pattern. Document these layers clearly and clearly indicate which components are

part of a layer and which aren't, this is extremely important for us in understanding how/where you see this pattern was implemented in the architecture.

- Docker has two types of plug-in architectures, one is for plug-ins inside the docker source code (like graphdrivers and logging drivers) and one for third-party plug-ins (the Volume and Networking plug-ins). Try to make a clear distinction which of the two you mention when you're talking about plug-ins.
- In your presentation you mentioned that there is a broker and that clients can use brokered authentication, and that there is a publish-subscribe mechanism for registry events, which are not yet documented here, but you are probably still working on that. Make sure to inspect the descriptions for these patterns well to see if these actually apply in the way you describe.

## Evaluation

- Be sure to think of a good method of measuring the impact of patterns. In your presentation you used unweighted arrows, but this may not work out as well because not every pattern has the same kind of impact on the key drivers they affect. Also, include the negative forces of these patterns where applicable.

Hopefully you will be able to make use of these points of feedback. Good luck on the final deliverable of this assignment!