R1: review of an article

Andrei Fangli, class 258
10th of November 2015

On The Liskov Substitution Principle

Published on objectmentor.com by Robert C. Martin, March 1996
http://www.objectmentor.com/resources/articles/lsp.pdf

The Liskov Substitution Principle is one of the SOLID[1] principles of Object Oriented Design. It is important to follow this principle at any point when one designs a class hierarchy. As illustrated in the paper not doing this breaks inheritance and derivatives cannot be used in places where the base class is used because as they are not extending their base class but changing it resulting in unexpected behaviour and the inheritance not being usable altogether.

In this article, Robert C. Martin dives into inheritance and illustrates common mistakes that are done while relying on class inheritance as a means to reuse code. Explains how Liskov Substitution Principle should be applied in order to create sound hierarchies and also makes a connection with the concept of Design by Contract exposed by Bertrand Meyer long ago. Last but not least sheds some light on the notorious “is-a” relationship that governs inheritance.

While the article exposes classic examples (e.g.: Square is not a Rectangle) that are thoroughly analysed and put against LSP[2], the author diverges at one point towards what makes a model valid. While this is helpful there is no clear statement on how to validate a model. Stating that analysing it in isolation is not enough to decide whether de design is flawed and that a model is considered valid in terms of its clients is not enough. There is no reference to another paper that researched the validity of models. This is not all, while the classic simplistic example of Square versus Rectangle receives such validation the “real world” example does not.

When the author finally reaches a critical point by exposing a “real world example” he fails to do the thorough analysis on this example as he did on the previous. Why is one approach compliant to LSP? Why the previous one where everything was encapsulated within a module is not? What was the convention for the respective module? How and when was it violated? These are all questions that remain unanswered and the reader must find the answers. While this is perfect from a didactic point of view it is bad from a research point of view. The author is not fully sharing his findings on the topic through all presented examples.

The first example which also happens to be the simples of all three (Square versus Rectangle, Persistent Set first approach and Persistent Set second approach) has a good thorough analysis with

[1] SOLID in computer science are a number of core principles that should be followed. They are guidelines meaning a software can be written well without fully following the principles every time, however any time this happens the one doing it should argument his/her decision.
[2] LSP stands for Liskov Substitution Principle.

hints towards Design by Contract and validating a design the others, which are also considered "real world examples", lack all of these. The author has somehow lost himself on the way.

The conclusion section is the most disappointing. It is mostly advertising and only the general conclusion, which also applies to SOLID principles, that LSP should be applied was stated. The results of other analysis done in the paper, such as the unravelled mystery of the "is-a" relationship, are not mentioned. The author started making a few references about Design by Contract but did not continue on that path a little further to expose how having it enforced by the language can help with design.