3.5.2014 Hannele Räsänen
Ohjelmatuotanto kevät 2014
Helsingin yliopisto

Summary: Is design dead? / Martin Fowler

This summary is about Martin Fowler’s article: Is design dead?

It seems that XP calls for the death of software design and even patterns are de-emphasized or downright ignored. In fact XP involves a lot of design. XP has rejuvenated the notion of evolutionary design. It provides new challenges and skills; simple design, refactoring to keep a design clean, and patterns.

In its common usage, evolutionary design is a disaster. The design ends up being a bunch of ad-hoc tactical decisions, which makes the code harder to alter. Planned Design is a counter to this. Designers think out the big issues in advance. They don't need to write code, they aren't building they are designing. They can use a design technique like the UML to work at a more abstract level. Once the design is done they can hand it off to a separate group to build.

Two of the greatest rallying cries in XP are the slogans "Do the Simplest Thing that Could Possibly Work" and "You Aren't Going to Need It". The way YAGNI is usually described is that, you shouldn't add any code today which will only be used by feature that is needed tomorrow. This sounds simple. XP's advice is that you don’t build flexible components and frameworks for the first case that needs that functionality. Let these structures grow as they are needed. Reasons; economics and that complex design is more difficult to understand than a simple design. What on Earth is Simplicity Anyway? The best advice came from Uncle Bob (Robert Martin). His advice was not to get too hung up about what the simplest design is. After all you can, should, and will refactor it later.

The relationship between patterns and XP is interesting, and it's a common question. Joshua Kerievsky argues that patterns are under-emphasized in XP and he makes the argument eloquently. But it's worth bearing in mind that for many people patterns seem in conflict to XP. My advice to XPers using patterns would be; Invest time in learning about patterns, concentrate on; when to apply the pattern, how to implement the pattern in its simplest form first and then add complexity later.

The point of YAGNI is that you don't add complexity that isn't needed. This is part of the practice of simple design. Refactoring is needed to keep the design as simple as you can, so you should refactor whenever you realize you can make things simpler. Can we use refactoring to deal with all design decisions, or are some issues so pervasive that they are difficult to add in later? Although there will be effort required to refactor the simple solution into what you actually need, this refactoring is likely to be less work than building all the questionable features.

So is Design Dead? Not by any means, but the nature of design has changed. XP design looks for the following skills; A constant desire to keep code as clear and simple as possible, Refactoring skills, to confidently make improvements whenever needed, A good knowledge of patterns, Designing with an eye to future changes, Knowing how to communicate the design to the people who need to understand it, using code, diagrams and above all: conversation.