

# Test Driven Development

A case study in introducing TDD to programmers

## Introduction

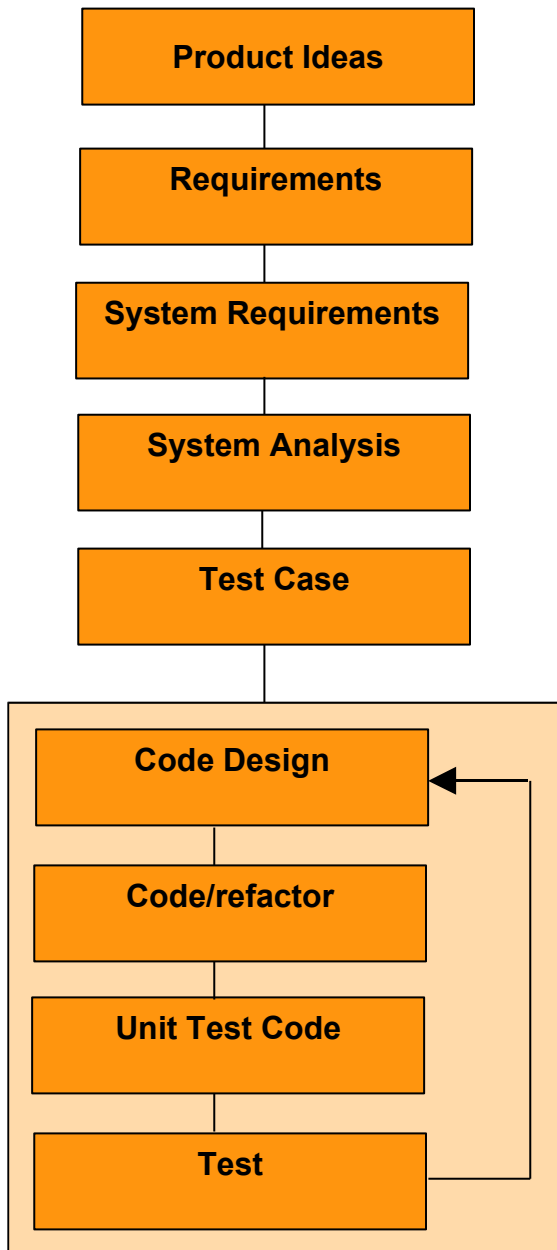
Pilz Ireland are currently evaluating the feasibility of introducing Test Driven Development. This document sets out to capture that process, and the arguments presented for and against within the organisation.

## Current Development Process

The current process used in Pilz stipulates that unit tests be written, with the goal of 100% code coverage. The applications been developed are Java GUI desktop applications, some particularly graphical, incorporating Java2D, SVG and OpenGL technologies. These applications, do not lend themselves easily to unit testing, and for this reason, unit tests have so far been almost the sole remit of model and controller functions. Developers in general write these unit tests after the functional code has been written, and the tests are written in such a way that they always pass.

The main benefit, from the developer's point of view, is that these tests capture breaks following refactoring or the introduction of new features, and are run nightly, with reports available every morning. They are used to verify code works as expected, not specification.

## Current Development Process



## Need for change.

The size and scope of applications being developed within the organisation has grown exponentially. This is the primary reason behind the search for an improved development process.

Combined with this, the recent move to developing products on the Eclipse Rich Client Platform, and using a plug-in architecture has brought with it a large increase in possible product permutations and as a result, increased the manual test burden. Most notably, the larger projects have brought with them longer requirement and analysis stages in the development cycle and an increase in the number of stakeholders inputting to these requirements.

Getting frequent and early feedback was key to handling these changes with minimum cost overrun. However, the larger projects have proven the existing process to be less suitable, as requirement changes are more costly than ever, due in part to the implications of using EMF and GMF (namely their complexity, and in GMF's case, the fact that it's such a new, and largely undocumented API), but also due to cross dependency of requirements through a number of features.

The new technologies, larger scope of projects and added managerial layers were proving a challenge for the existing process. Appropriate requirement specification management and quality control were essential.

## Introducing Test Driven Development

With the increase in the number of stakeholders, the complexity of meeting requirements needed to be managed. Improved communications is always an on going process, and a developer lead initiative to evaluate Test Driven Development is under way to pilot it for suitability and effectiveness within our organisation, as a communication and development tool.

The benefits of Test Driven Development are well documented. A list may include:

- Test becomes more design than validation.
- Encourages small, manageable development steps.
- More productive, as failed tests are easier to fix, and found sooner in the development cycle.
- TDD gives rapid feedback to both development and other stakeholders.
- Supports looser coupling of code and cleaner interfaces (improved design).
- Complimentary to agile model driven development.

With the benefits clear, how does a Test Driven Development process get introduced to an organisations development cycle?

In Pilz, it was driven by a technological evangelist, touting the benefits after spending an elaboration trying out this new development technique.

To get everyone on board<sup>1</sup>, management and development alike, the developer in question organised a presentation to sell the merits and stamp out the myths surrounding it. The fact that he was speaking from positive experience gave much weight to the argument (as opposed to reading any given article would).

However, the myths, although in theory squashed, were to materialise quite early on in the evaluation.

## First impressions of Test Driven Development

- Like pair programming, many initially thought Test Driven Development was a luxury item, afforded to few organisations in practice. There was now more code to write, maintain and a larger delay in displaying visual results to requirement stakeholders for that essential early feedback.
- Changes in requirements would mean first changing tests, then functional code. This overhead was time costly, and, with tight deadlines always looming, it was beginning to appear to some that this new test driven approach to development was not for this organisation.
- The GMF auto generated code, and other auto generated code threw up an unexpected situation for this new approach. How can a developer write tests first in this circumstance? And was it worth it in this case.<sup>2</sup>
- Within our organisation, developers are actively involved during the requirement and analysis phases. The advantage of encapsulating requirement understanding was not a deficit thought in need of addressing.
- A sizable portion of code is at the user interface level. Test Driven Development does not capture workflows, usability, or applications meeting their graphical design specifications.<sup>3</sup>

---

<sup>1</sup> A necessity when recommending such a relatively dramatic change to peoples work practices.

<sup>2</sup> Auto test generation may solve this issue.

<sup>3</sup> Nor does it claim to be the sole method of testing, and can still work with manual and automated GUI testing.

## Second Impressions

Once a full elaboration was complete, the experience was very positive. The overall impression was:

- Implementations were delivered in faster time.
- More time was spent on design than debugging.
- Implementation code was cleaner.
- Developers were more confident making changes and refactoring code
- And once tests were continuously run and maintained, they worked well in giving immediate regression testing results that were easy and fast to address.

The principle feedback from this new approach is that Test Driven Development is a strong design tool that saves development time and contributes to higher quality code.

## Conclusion

The roll out of Test Driven Development is on going, and some flexibility to the development approach has been taken with regard to user interface code. *(Having said that, it is already encouraging a lighter UI layer with more logic pushed to the controller level).*

Test Driven development in Pilz Ireland is leading to development with further emphasis on requirements and design.