



# Iterative Development Research

26/09/2014

**Charlotte Hutchinson**

CO600: JustHealth
Supervisor: Yang He

[Version 1.2]
**Last saved by: Ben McGregor**
**Last saved on:** 29/03/2015

# Contents

# 1.0 Introduction

JustHealth have elected to use an iterative development approach for producing our application and website. Our iterative approach allows us to combine different aspects from other methodologies to create our iterative approach. Having short development cycles allows us to gain continuing feedback and quickly come up with an overall plan that is expected to evolve through the life of the project (Beck, 2004). It has the objective approach of producing working software and taking precedence over exhaustive documentation, as per the manifesto (Appendix 1).

Research conducted The Standish Group has shown that possibly three times as many projects fail when the waterfall method is utilised when compared with those using agile methods (Standish Group, 2011).

# 2.0 Analysis

All of the JustHealth project group have experience in different development methodologies including waterfall, agile and a general iterative approach. The reason we have chosen to use an iterative approach is to combine ideas and strengths from different methodologies and use them to simplify the way we manage application development. We feel this is superior to the traditional methods, in most cases, particularly those involving small projects with a small, close knit development team like ours.

The iterative approach also perfectly suits the iteration requirements of the project, such that we have to produce working software to present every 2-3 weeks. It also suits the project having an evolutionary design process that lasts as long as the system lasts (Beck 2004). Below are other practices or agile and extreme programming (XP) we are using in our iterative approach:

| **Practise** | **Benefit** |
|---|---|
| Pair Programming | Pair programming has been shown to reduce the number of errors in code by 15%. It also has the added benefit of increasing developer satisfaction and designs. (Williams et al, 2000) |
| Test Driven Development | Having easily repeatable and automated tests written before functionality is implemented allows us to easily ensure that our functionality works and is of an appropriate standard. By focusing on writing code to pass those tests we are encouraged to write cleaner code that implements our designs. (Beck, 2002) |
| Triweekly Team Meetings | Keeping the developers in close contact allows us to keep on track and focused as best as possible. Here we can examine the work that has been done, what we are going to doing |

| Practise | Benefit |
| --- | --- |
| | moving forward and discuss anything that is hindering our progress. |
| Collective Code Ownership | With all team members actively contributing and owning all of the project code and documentation we can ensure that issues are much more readily spotted and fixed, and that each developer has another who is ensuring that they are sticking to the agreed practises and methods. |

# 3.0 References

Mike Cohn (2012). *Agile Succeeds Three Times More Often Than Waterfall*. [Online] Available at: http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall [Last accessed: 07 October 2014]

The Standish Group, (2011). *Chaos Manifesto*. [Online] Available at: http://www.versionone.com/assets/img/files/ChaosManifest_2011.pdf [Last accessed: 07 October 2014].

Beck, K (2004) *Extreme Programming Explained* - Embrace change

Williams, L.; Kessler, R.; Cunningham, W.; Jeffries, R. (2000). *Strengthening the Case for Pair Programming*. [Online] Available at: http://collaboration.csc.ncsu.edu/laurie/Papers/ieeeSoftware.PDF [Last accessed: 07 October 2014].

Beck, K (2002). *Test Driven Development by Example*. Addison-Wesley.

Beck, K. et. al (2001) *'Manifesto for Agile Software Development'*. [Online] Available at: http://agilemanifesto.org/ [Last accessed: 07 October 2014]

# 4.0 Appendix 1

## Manifesto

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

| | | |
|---|---|---|
| Kent Beck | James Grenning | Robert C. Martin |
| Mike Beedle | Jim Highsmith | Steve Mellor |
| Arie van Bennekum | Andrew Hunt | Ken Schwaber |
| Alistair Cockburn | Ron Jeffries | Jeff Sutherland |
| Ward Cunningham | Jon Kern | Dave Thomas |
| Martin Fowler | Brian Marick | |