Notes: This document will need to be downloaded and saved to github regularly.

* How often?
* Who will be responsible?

# Introduction

## Purpose

Independent developers [startup name] have displayed a viable game development concept as part of an international contest. As part of a government initiative to support growth of the game industry in [this region], an award of local and national government funding has been granted to [startup name] as a startup fund. Prompted by the need to develop a quality product under a limited budget and with the New Zealand Public as significant stakeholders; this study was commissioned to recommend the best software development approach. The report covers an explanation of several suitable approaches, their advantages, disadvantages, risks and benefits; and recommends a particular approach as the most suitable for this product.

The report as-delivered also includes a Practice Guide, as well as Tool Guides and a Learning Package. These additional artifacts are provided to enable the [startup name] to put the recommended approach into practice.

## Product Context

### Scope

[game name] is projected as primarily a PC (Windows) game, although other platforms such as linux, iOS and Android are also considered. The project lifecycle has been initially scoped for two years to market release and the workforce is anticipated to grow from a current team of four into a team of 45 - 50 people. This indicates a highly dynamic and fluid environment, requiring robust change management and communication.

The game concept, though juvenile, is already formed and has been accepted as the premise for government funding. Development is expected to adhere to this concept as far as reasonably practicable and any approach which risks deviation or scope-creep must include processes to mitigate these phenomena.

### Stakeholders

[startup name] will be accountable to the New Zealand public and must be able to show positive progress throughout development. It is expected that proof of progress will need to be provided at regular intervals. As of the writing of this report, New Zealand Government has not issued any regulations for game development, other than publication classification requirements (New Zealand Legislation, n.d.).

* Criticality
* Linear or sandboxed

## Agile vs Plan-driven Development

There are considered to be two main philosophies which drive software development - ‘traditional,’ plan-based development and agile development (Boehm & Turner, 2009). Both ideologies have their merits and drawbacks, described by Boehm and Turner as the difference between discipline and rigour of process and project adaptability respectively. Any software development project must assess the suitability for either of these main concepts and determine if one, the other or a mixture of both are required.

Plan-driven development evolved out of environments such as aerospace and commercial industries where a focus on documentation and processes and a change-averse environment are valued (Boehm & Turner, 2009). The advantage of this philosophy is an holistic plan for and view of a complete project and its expected outcomes - plan-driven development is predictable and can be bound by regulations and industry standards.

Comparatively, the agile philosophy eschews an upfront plan and makes smaller, feature-based, working modules of a product which is ultimately developed over several iterations (Meyer, 2014). Agile development environments are change-aware and are inherently more adaptable to evolving requirements and shifting goalposts.

In the case of [this study], an agile approach is favoured for the following reasons:

* Game development is not regulated,
* Dynamic development environment,
* Product criticality is low,
* No legacy code replacement/integration, and
* Product is not likely to require integration with external/custom systems

However, the projected size of the development team is not considered optimal for a fully agile approach. Boehm and Turner (2009) describe several scenarios where teams of 50 - 150, although successful, were not easily managed. It is for this reason the study does not completely rule out aspects of plan-driven development which may be beneficial in this case.

# Agile Development Approaches

This section outlines a collection of suitable development approaches, each drawing on the Agile philosophies of [insert philosophies with references].

## Crystal [Sez researching Crystal]

### Overview

The Crystal ‘family’ of methodologies are intended to provide project outlines which guide rather than dictate the requirements for an agile project. Each crystal type is identified by a colour, beginning at Clear and ranging through Yellow and Orange to Red, which indicate increased project requirements as a function of increased project size and/or criticality (Cockburn, 2007). The methodology aims to be as light as possible for any given project and requirements (properties) are tailored accordingly. All colours in the Crystal family support the following values and principles:

Values: 1) The techniques and practices used will be people-centric and communication-centric, thereby supporting the human element; 2) the design environment will be highly tolerant, recognising the varied nature of human cultures (Cockburn, 2007).

Principles: The following list is taken directly from Cockburn (2007, p.182):

1. Interactive …
2. Excess…
3. Larger teams…
4. Greater ceremony…
5. Increasing feedback…
6. Discipline, skills…
7. Efficiency is expendable…

### According to Meyer (2014), small projects can utilise Crystal Clear while Orange is older and suits larger projects. Cockburn (2007) states Crystal Orange was developed for a team of up to 40 people, where the project length is expected to be one to two years and the system is not life- or legally critical. In the case of [project name], the most suitable type of Crystal project is likely to be Orange. Cockburn (2007) also describes the method as suitable for projects co-located in one building and where time-to-market is important. These are the characteristics of the [game name] project.

### Practices and Requirements

The Crystal methodology does not prescribe specific techniques or processes, instead; it provides a collection of project roles and practices which encourage a minimum overhead while ensuring completeness of project design. Specifically; Cockburn (2007) instructs that Crystal Orange requires the following project standards to be met:

* Incremental software delivery (three - four months)
* Milestones track progress using deliveries or decisions
* Functional regression testing is automated to some degree
* Direct user involvement
* ‘...two user viewings per release
* Downstream activities start as soon as upstream is “stable enough to review”
* Product- and methodology-tuning workshops are held at the start and middle of each increment.’ (p. 340)

Cockburn (2007) instructs that the processes for achieving these standards are chosen and defined during the initial ‘method-tuning’ stage of project design. Crystal allows for processes from other methodologies to be utilised, so long as they fulfil the requirements of the standards (Cockburn, 2007). In addition to the standards, Cockburn also dictates the need for fourteen roles within the project, including Sponsor, Business Expert, Project Manager, Architect, and various designers, programmers and testers among others.

### Advantages and Payoff

### Disadvantages and Risk

### Suitability

Reference Meyer, a few of the practices in Crystal are suggested to be hype.

# Recommended Practice

According to Boehm and Turner (2009), “Analyzing the project risks is an effective way to help managers and practicioners determine the right weight of process.”

and

“Asking ‘*Is it riskier for me to apply (more of) this process component or to refrain from applying it?*’ and honestly evaluating the answer can lead to the definition of reasonable, practical and effective hybrids that balance discipline and agility.”

# Practice Guide

## Communication and Collaboration

## Requirements Engineering

## Planning and Tracking

## Change Management

## Quality Assurance

# Tool Guide

## Communication and Collaboration

## Requirements Engineering

## Planning and Tracking

## Change Management

## Quality Assurance

# Learning Package

## Goals

## Target Audience

## Plan

## Required Resources

## Activity

# Executive Summary

# Retrospectives

## Team Retrospective

## Individual Retrospective

### Gabby

### Sarah

### Wayne

# Bibliography

Boehm, B. & Turner, R. (2009). Balancing agility and discipline. Addison-Wesley.

Cockburn, A. (2007). Agile Software Development. Addison-Wesley.

Meyer, B. (2014). Agile!. Zurich, Switzerland: Springer.

New Zealand Legislation. (n.d.) Retreived from http://www.legislation.govt.nz/