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

* How often?
* Who will be responsible?

# Introduction

## Purpose

Independent developers Saturn Games 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 Canterbury an award of local and national government funding has been granted to Saturn Games 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 Saturn Games to put the recommended approach into practice.

## Product Context

### Scope

The game itself, *Gibbous Earth*, 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 within the first three to four months. Approximately 40% of the workforce will be dedicated to software development/quality assurance, indicating a development team of 20-25 people. This growth 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 honour the initial concept, however; there is scope for the product to evolve as team members provide their experience and ideas.

### Stakeholders

Saturn Games shareholders consist of the four founding staff members, and an additional three active investors who were also involved in the funding application. Saturn Games 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 be provided at approximately six monthly intervals. As of the writing of this report it is assumed, through lack of evidence to the contrary, that New Zealand Government has not issued any regulations for game development, other than publication classification requirements (New Zealand Legislation, n.d.).

### Project Criticality

The main risk of the project is loss of investment, and there are no life-critical or financial security interests. There is no risk of loss-of-contract if project milestones are not reached. The project criticality may therefore be considered to be low.

## 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 the *Gibbous Earth* project, an agile approach is favoured for the following reasons:

* Game development is not regulated,
* Dynamic development environment,
* Scope to add, remove and change features during development,
* Product criticality is low,
* No legacy code replacement/integration, and
* Product is not likely to require integration with external/custom systems.

It is worth noting that Boehm and Turner (2009) describe several scenarios where teams of 50 - 150, although successful, were not easily managed. Although Saturn Games expects to employ up to 50 people, the development team is likely to consist of 20-25 employees. For this reason GWS advise an Agile-based approach.

# Crystal Development Approach

This section outlines Crystal, a development approach which meets the Agile philosophies developed by Beck, et al. (2001) such as:

1. Early and frequent delivery of working software which brings value,
2. Support for requirement-change,
3. Daily integration of Business & Development,
4. Support and trust of employees,
5. Maximisation of face-to-face communication,
6. Sustainable pace of development,
7. Reflection and adjustment of the development environment,
8. Efficiency through simplicity.

## 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, with one main concept: “Osmotic Communication” (Cockburn, 2007; Meyer, 2014). Cockburn explains this is when team members work so closely they gather valuable information from conversations and work stations which occur naturally in the work environment.

All colours in the Crystal family support the following values:

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

These values will be essential to the *Gibbous Moon* project as the team will be newly established and will likely go through forming and storming periods before norming and performing. Crystal offers a pathway to a supportive development environment, tolerant of change and the people who implement it.

### According to Meyer (2014), small projects (teams up to eight) 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 the *Gibbous Earth* project, the most suitable type of Crystal project is likely to be Yellow, which is based on an expansion of Clear and is for teams of size fifteen to 30 (Cockburn, 2007).

## 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 Clear requires the following project standards to be met:

* Incremental software delivery (two - three 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). For example, Cockburn explains that Extreme Programming (XP) meets all the requirements of Crystal Clear except for Clear’s requirement for documentation. In addition to the standards, Cockburn also dictates the need for four to fourteen roles within the project, depending on the project ‘colour’. For Crystal Clear these would include Sponsor, Senior and standard designer-programmers and the user (Cockburn, 2007). If Crystal Yellow is to be selected as the project approach, Saturn Games must ‘method-tune’ Crystal Clear it the beginning of the project and incorporate elements of Crystal Orange as the team sees fit.

## Advantages

The main advantage of the Crystal methodology is that it is designed to be fine-tuned to fit the project needs. By ‘stretching’ the lighter version up for a larger team (i.e. from Clear to Yellow) Saturn Games can be sure they are not over-burdening the development team with excess managerial overhead. Regular reflection and tuning throughout the project life enables the development environment to constantly evolve, thus coping with the rapid increase in employee numbers and the forming/storming periods associated with new teams.

## Disadvantages

By Cockburn’s own admission, the lack of prescribed processes in Crystal makes it daunting for beginners in Agile development. To make the most of the approach; the development team should have strong experience in methods such as Scrum, XP, Kanban and others in order to tune Crystal to meet the needs of the *Gibbous Earth* project. This lack of prescription would require greater pre-development design, although once up and running Crystal’s tolerance of change will make it easily manageable on a day-to-day or week-to-week basis. Meyer (2014) notes the requirement to change to a new method if the size or the criticality of the project changes significantly, however; an experienced team will handle the shift smoothly, noting there is potential for some significant burden in re-tuning or changing methods for gross project changes.

## Suitability

Meyer (2014) suggests some of the practices or techniques encouraged by Crystal are overhyped, such as pair programming, sustainable pace, and open-space offices. There is no requirement to utilise these techniques if they do not suit the Saturn Games team. Once again, with a high level of Agile experience on the *Gibbous Earth* project, Crystal Clear/Yellow would provide suitable guidelines for creating an ad-hoc approach which still meets the minimum requirements for a quality product delivered from within a dynamic and safe environment.

# Practice Guide [Sez]

## Communication and Collaboration

Practices which will need TOOLS:

* **Information Radiators**
  + eg planning boards. Have a look at this: <https://youtu.be/eZpotzEeF8E> see the mix of projector/burndown and physical. A write up on it is here: <http://www.prodataconsult.com/about-us/best-practice/view/27/78/possibly-the-best-in-denmark/>
* **Continuous Integration**
  + I will be stating the team will need to keep their code in a shared repository which can handle branching and merging of code. I think both git and mercurial can handle this.

1. Osmotic / co-located / same floor
2. Cross-discipline (user, client, devs, testers, BAs, executives)
3. Direct user involvement
4. Tuning workshops
5. Information Radiator

## Requirements Engineering

Practices which will need TOOLS:

* **Use-cases or feature descriptions**
  + Remember the tools don’t have to be digital/online

“Annotated Use-cases or feature descriptions”

## Planning and Tracking

Planning practices:

* **Exploratory 360o**
* **Methodology Tuning**
* **Project Map & Release Plan**
  + Tied to Project Status
* **Iteration Plan**
  + Probably XP’s ***Planning Game*.** You should have access to [this](http://link.springer.com/chapter/10.1007%2F3-540-44870-5_14), if you search for it through AUT library, and there is [this](http://hacknplan.com) too. Don’t know how useful they are but it might get you started.
  + Tied to Iteration Status
  + Dynamic Priority List (e.g. Product Backlog)

Tracking Practices

* **Project Status**
  + Tied to Project Plan
* **Iteration Status**
  + Tied to Iteration Plan
* Burn charts

*Documentation is required but how/what is determined by the team.*

7. Milestones

8. Common Object Model

9. Screen drafts

10. User viewings - scheduled

11. Early Victory / Walking Skeleton

12. Blitz Planning

13. Planning Game

14 Burn charts

15. Dynamic Priority List

16. Release Sequence

## Change Management

16. Product Tuning / Methodology Tuning

* **Reflection Workshops**

18. **Version Control** - probably git or mecurial.

17. - there is something called ‘feature thrashing’ where iteration goals receive conflicting priorities. See Fig 5-22, Cockburn 2004?? *Crystal Clear*

18. Reflection Workshops

## Quality Assurance

18. **Test Driven Development**

19. **Automated Testing**

20. Templates for work products

21. Standards for coding & user interface

22. Standards and details of regression testing

# Tool Guide [Gabby]

## Communication and Collaboration

## Requirements Engineering

## Planning and Tracking

Note to Gabby: Crystal states that: “The most important tools the team can own … are these: A version and configuration management system [and] a printing whiteboard.” Git would be a great example of the first, and these days we can just take photos of the white board!

## Change Management

## Quality Assurance

# Learning Package

## Goals

What are going to learn?

## Target Audience

Who is going to do the learning? & how much experience do they have?

## Plan

How will they learn, how will the tutorial or teaching be accomplished?

## Required Resources

## Activity

Steps for them to practice and maybe questions for them to answer or observations to make

# Executive Summary

# Retrospectives

## Team Retrospective

## Individual Retrospective

### Gabby

### Sarah

### Wayne

# Bibliography

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). *Principles behind the Agile Manifesto*. Retrieved from http://www.agilemanifesto.org/principles.html

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/