# Practice Guide

This guide is designed to describe various Crystal practices in more detail. They are grouped into functional areas which Saturn Games will need to address as they plan and build the *Gibbous Earth* development environment.

## Communication and Collaboration

The two main practices which facilitate effective communication are Osmotic Communication and Information Radiators. Direct User Involvement may also be considered as a communication method, however; it is discussed in more detail in the Quality Assurance section of this guide. Collaboration practices which are outlined here are continuous integration and the use of cross-discipline teams.

### Osmotic Communication

As discussed in the report, osmotic communication is the transferral of ideas and information via overhead conversation. To provide more detail to the reader, it also includes the practice of removing barriers to conversation (such as office walls, doors and flights of stairs, etc.); thus reducing the likelihood and impact of broken trains of thought.

The *Gibbous Earth* project is too large to achieve co-located team members across the entire project environment, therefore; it is recommended the main project team be split into smaller, discipline sub-teams which are each responsible for different aspects of game development. Each sub-team may then achieve co-location and will benefit from osmotic communication, and ideally the entire project will be located on the same floor. The coordinator may need to experiment to determine whether the team leads should be located with their team or in a group together - it will depend on which flow of information brings the most value to the product, and whether barriers to communication are supportive (allow the expert to get the work done) or a hindrance (junior members do not benefit from having an expert in earshot (Cockburn, 2005). For more detail and understanding of this practice, refer Cockburn 2005, 2007).

### Information Radiators

It is important in a large and dynamic environment that every team member has easy access to project goals and status, and other relevant information. Information radiators typically consist of large planning boards placed within view of the development team. Because people are able to see the information without having to ask questions, more information can be transferred with fewer interruptions (Cockburn, 2005). All team members have shared ownership of the board, and are responsible for keeping its contents up to date. Although the contents of the information radiator will change as the project is tuned during reflection workshops, to begin with the *Gibbous Earth* project should ‘radiate’ the following information (see Cockburn, 2005 p. 54):

* Results of the most recent reflection workshop,
* Overview of the domain,
* Server statuses,
* Status of the current iteration including the work set and assignments,
* Status of tests written and/or delivered,
* Successes such as the number of use cases delivered.

For more information and implementation instructions, refer the tool guide which accompanies this report.

### Continuous Integration

Continuous integration (CI) requires a shared repository to hold the combined code and the ability to make local modifications which a) do not affect the main branch of code and b) can be merged into the main branch after changes have been reviewed. This can be achieved with the use of a versioning and integration tool such as Git or Mercurial, combined with either a proprietary repository held by Saturn Games or an online repository such as Github or Atlassian’s Bitbucket.

GWS recommends the *Gibbous Earth* project aim to integrate several times a day in order to minimise integration issues. Such issues will include mistakes, build-up of misunderstandings/additional errors and the need to search significant quantities of code to find the problem (Cockburn, 2005). However; hourly integration may not be achieved - in which case Cockburn (2005) advises daily integration is acceptable, but; integration should not be less frequent than every second day. By having the team aim for hourly iterations, integration will be kept at the forefront of the project’s shared consciousness. If integration encounters significant issues on a routine basis, the first solution is to integrate more often. GWS has provided more details on CI tools in the tool guide.

## Requirements Engineering

The focus on embracing change requires Agile methods to manage changing requirements throughout the development process, thus less import is placed on heavy, upfront requirement-elicitation techniques. Agile methods such as Crystal focus more on gaining understanding through face-to-face communication which leaves requirement-based project artifacts to serve more as a reminder to have a conversation with the user/client than to act as an explicit direction.

### Actor-Goal List

The combined responsibility of the Business Expert and Expert User, the actor-goal list provides an overview of the system and its high-level functional requirements (Cockburn, 2005). The actor is a type of user (e.g. experienced gamer, child, team member, etc.) who will have major goals associated with their use of the system/game (e.g. create a lobby, set mele rules, team-play with friends, load solo campaign, etc.). Cockburn (2005) advises the goals can be used as a skeleton for the project status. GWS recommends goals to also be adapted for use as delivery milestones, as described in the Communication and Collaboration section of this guide.

### Use-cases

To support the goals of each actor, a use-case is a successful scenario which describes how the actor will achieve the goal, for example:

**Multiplayer game**

“The initiating player logs on to the game and creates a new lobby. She invites three friends to join the lobby. Her friends accept the invitation and join her in the lobby.

At any time after the lobby is created and before the game countdown initiates game-play, the initiating player can cancel the game and all players will leave the lobby. At any time after joining the game, an invited friend can leave the lobby.

The Crystal Clear format suggests a successful scenario for the first part, and failure handling in the second (Cockburn, 2005). Since *Gibbous Earth* has the advantage of flexibility in requirements (compared with projects with high criticality or integration requirements) the use-cases can remain relatively conceptual, leaving the project team and sub-teams to discuss and explore during development and iteration planning. Cockburn (2005), Chapter 5 gives more detailed instructions for use-cases in Crystal Projects.

## 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

Any Agile project aims to embrace change (Beck, et al., 2001) and the Crystal family is no exception. By embracing change, Saturn Games will be in a strong position to take advantage of any competitive edge which may arise during the course of development. Short iterations and short delivery cycles enable the project to pivot during development as and when necessary. Embracing change does not, however; mean a project should be allowed to fall victim to loss of direction or value due to unmanaged change. When change is considered for incorporation it must be prioritised against the current status and a decision made on what value will be added or reduced.

Requirements change has been discussed as part of the Requirements Engineering section of this guide, this section details management of changes to the product and changes to the project itself.

### Reflection Workshops

One of the twelve Agile principles, reflection allows the team to continuously improve and become more effective (Beck, et al., 2001). Crystal Clear advocates the use of the “Keep-try” Reflection workshop and this simple one-hour technique allows the team to self-direct; choosing to keep strategies which work, discard those which don’t and try new ideas in the next iteration (Cockburn, 2005). According to Cockburn the workshop is a pause in development which shall take place after every delivery and may also be held mid-way through the delivery cycle, after an iteration. The full frequency of reflection workshops is left to the development team to decide - although this decision, too, ought to be included in the reflection (Cockburn, 2005). Cockburn also explains it is most likely there will be many workshops and much to discuss while the project is in its infancy but they will drop away as time and experience bring stability.

Keep-try requires a poster or whiteboard to be separated into three areas, “What we should keep,” “Where we are having ongoing problems,” and “What we want to try in the next time period.” (Cockburn, 2005, p.65-66). The important points for running a Keep-try workshop, according to Cockburn, 2005) is to limit time spent on the problem area and maximise the time spent on deciding what to try next. By keeping this focus in a reflection workshop, the *Gibbous Earth* team will move forwards with positive, problem-solving change rather than become bogged in the weeds of project issues. For more information on the tools required to carry out a Reflection Workshop refer to the Tool Guide which accompanies this report, or review Cockburn (2005), Chapter 3.

### Version Control

Like continuous integration, version control is best conducted with the use of a digital tool and a central repository. Along with documentation (also required by the Crystal family of methods), version control should be able to tell a story of the product development. This enables project reviews to determine how and why changes were made, in turn aiding future development and maintenance. While documentation can be as simple as photos of diagrams made in planning sessions, version control must be more robust. GWS recommend either Git or Mercurial, which can be used for both version control and to facilitate continuous integration. All developers shall have access to the repository and commit their changes at regular intervals. The tool guide provides more direction on the use of a version control tool such as Git.

## Quality Assurance

18. **Test Driven Development**

19. **Automated Testing**

**Direct User contact**

20. Templates for work products

21. Standards for coding & user interface

22. Standards and details of regression testing

# Bibliography

Please refer to the bibliography in the GWS team wiki at https://autonline.aut.ac.nz.