|  |  |
| --- | --- |
| XP Process Review | September 24  2011 |
| This document contains notes on Team Sneaker's implementation of Extreme Programming. | Craig Hokanson  George Dean  Jason Leng  Sion Chaudhuri |

# Contents

Contents 2

Introduction 3

Iteration 0 Summary 3

Iteration 1 Summary 3

Iteration 2 Summary 4

Iteration 0 Practices 4

Whole Team 4

Stories 5

Quarterly Cycle (Release Planning) 5

Iteration 1 Practices 6

Informative Workspaces 6

Pair Programming 6

Weekly Cycle 7

Test-First Programming 7

Continuous Integration 8

Iteration 2 Practices 8

Ten-Minute Build 8

Incremental Design 9

Root-Cause Analysis 9

Iteration 3 Practices 10

Shared Code 10

Sit Together 10

Slack 10

Energized Work 11

# Introduction

## Iteration 0 Summary

Team Sneakers has identified three XP primary practices that will help it become successful. The practice of “Whole Team” helps us to collaborate effectively, thinking about each remote member. The second practice of using “Stories” focuses our development, and helps us know what goals to accomplish. The last XP practice “Quarterly Planning” has been effective, as we have identified the scope of this project, and the methods for achieving that.

## Iteration 1 Summary

Team Sneakers has spent a significant amount of time trying to follow many of the available XP practices. Through the course of this iteration, we have focused on creating our informative workspaces, adhering to pair programming principles, and conducting effective weekly cycle meetings. We have had much success as we have learned more about these XP practices, and we feel more confident as a team because of them.

Since we are not required to institute a test-first approach in this iteration, we have not focused on this practice, but we have tried to follow it when possible. The practice of creating tests has helped us as we have continually integrated our code to GitHub, by giving us confidence that our code won’t break.

## Iteration 2 Summary

In this iteration, we have focused on three XP practices: Ten-minute builds, Incremental design, and Root-cause analysis. The use of a continuous integration server, and the limited size of our project has made our builds much faster than 10 minutes, though there has been a few small issue in getting set up with selenium for integration testing.

In the true spirit of XP, we have also identified root-causes in the defects that we find, and use incremental design when necessary to fix those issues. These practices have helped us move forward with confidence in creating an exceptional application.

# Iteration 0 Practices

## Whole Team

Initially this practice was categorized as not applicable by our team; however, through a focused review, we’ve found that with some modifications it does fit our needs. Team Sneakers strives to keep an attitude of sharing and openness, our having all documents up on Google docs, where anyone on the team can access them and modify them, reflects this. We also set up a Google Calendar where we placed our availability, allowing all team members a view of the whole teams availability. We also schedule all our pair sessions, making sure to rotate with each other so everyone pairs together throughout the week.

Although each member of team Sneakers comes from a software engineering background, we currently work in different business sectors, which helps us bring different perspectives, ideas and technical expertise to the project.

## Stories

Team Sneakers utilizes an online system called Pivotal Tracker for managing our story cards. This enables us to effectively and collaboratively create and modify our stories, so they can represent a single piece of customer-visible functionality. With this tool, we can record the story title, description, acceptance tests, status, type, and associated tasks. For each story card we have set priorities, estimated difficulty, and performed preliminary assignments to iterations.

## Quarterly Cycle (Release Planning)

For this iteration we have worked towards a successful quarter in three ways. First, we have written stories that will guide and focus our development during the upcoming quarter. Second, we identified a subset of the Extreme Programming practices, which if followed, would help us be an effective team. In doing that, we looked at which practices would need to be modified for our remote team. Third, we have discussed a theme for this quarter, and have talked about how to incorporate that theme into our project development.

# Iteration 1 Practices

## Informative Workspaces

The informative workspace practice encourages the team to make our workspace about our work, so that an interested observer could take in the state of the project within about 15 seconds. Since we are a remote team, this has been a challenging practice. Our best successes come from developing virtual workspaces with Google Docs and Pivotal Tracker. In Google Docs, we are able to use a shared collection, which provides the entire team with up-to-date documents that serve as a replacement for a physical bulletin board or whiteboard.

Using Pivotal Tracker, we are able to make the progress on our story cards clearly visible, so that anyone who glanced at the screen would immediately gain a clear understanding of the state of the project. We also meet at Sion's once a week for 4 hours. There we are able to create temporary informative workspaces through the use of a whiteboard and big screen.

## Pair Programming

Pair programming is one of the key practices that we use. This benefits out team greatly, as it enables each team member to become familiar with the entire project, and increases his individual knowledge about a specific topic. We have set up a “pair session” calendar, where we have outlined all of our pair sessions, so we know who will be pairing with whom for any given day.

Since we are a remote team, we are largely unable to sit at the same physical machine; despite this, we have followed the spirit of this practice entirely. We have employed remote collaboration tools such as TeamViewer, Adobe Connect, Skype, and Google Voice to simulate the experience of sitting together at a single machine. We have written every single line of our production code using pair programming.

## Weekly Cycle

The Extreme Programming practice of weekly cycle states that work should be planned a week at a time. We follow this process by beginning each week with a planning and reflection meeting.

We begin this meeting with reflection. We review story cards that were completed during the past week and compare our actual progress with our expected progress; we use this information to calculate the team velocity. Then, we seek to identify what went well and what could be improved for the future.

Next, we plan for the upcoming week. We discuss the results of the pair sessions and resolve any remaining questions. Then, we discuss the status of the deliverables and identify plans for future pairings. Finally, we identify action items for individuals to take on during the coming week.

## Test-First Programming

The practice of test-first programming encourages programmers to write automated tests before production code; in this way, the failing test becomes the motivation for development. Since test-first programming is not a formal requirement until the second half of the semester, our team has not been strict about it and we do not have complete code coverage. However, we have developed a substantial portion of our codebase using test-first programming, and we are becoming much more familiar with the workflow over time.

## Continuous Integration

We believe, that by running our tests often and frequently checking code in, we will mitigate any kind of problems that might arise. Our goal has been to fully check in all code after a block of pair programming time, with frequent local commits in between. This ensures our builds work at any given time, and it ensures confidence that the code is ready for the next pair session.

Also, each push causes the continuous integration server (Goldberg) to run our entire suite of tests. In the event that we encounter a problem, an email notification is sent out and we fix the issue as soon as possible.

# Iteration 2 Practices

## Ten-Minute Build

Due to the limited size of our project, our team has not encountered any problems in maintaining a test suite that runs in under ten minutes. We have not had to resort to two-stage testing, nor has it been necessary to optimize the performance of the test suite.

The greatest threat to the ten minute build practice is integration testing. One of our team members works in a Windows development environment, which means that we must use the full-head Selenium webdriver instead of the much faster, headless (but unsupported) capybara-webkit. Moreover, Selenium itself runs substantially slower on Windows than it does on Mac OS X. Even though we have identified this as a potential threat, we feel confident that if it becomes a problem, there are other recourses for moving forward, such as spork for preloading the rails environment

## Incremental Design

The XP practice of incremental design refers to the decision to perform just-in-time design, instead of big-design-up-front. Our team follows the practice of incremental design by revising our design decisions over time to reflect changes in our understanding. Many times we have stopped our progression through a task card, and decided that it would be better to take a few minutes and refactor the code so it would be cleaner and clearer.

Also, as we have continued in this application, different requests for functionality have been introduced. These changes drive an incremental approach to design as we restructure and refactor the code, and talk about how to implement the new features.

## Root-Cause Analysis

This XP practice drives the search for what went wrong, and promotes fixing it so it won’t happen again. Team Sneakers uses this during our development, and when we identify a problem, we write the test that matches that failed functionality, and then write code to make that test pass.

During the first iteration, our team found that we ran out of time during the iteration-planning meeting. As a result of this, we had to replace a pairing session the next day with a continuation of the meeting. When we didn’t finish our planning during that session, we were forced to complete the meeting during a third time slot.

Instead of passing this problem off as a one-time issue, our team identified the root cause: we had fundamentally underestimated the amount of time required to plan an iteration. Armed with this understanding, we went on to plan larger time blocks for our remaining iteration planning meetings. During this iteration, we were able to complete our planning meeting in the time slot that we had allotted.

# Iteration 3 Practices

## Shared Code

*<provide evidence on how the team is following this practice>*

## Sit Together

*<provide evidence on how the team is following this practice>*

## Slack

*<provide evidence on how the team is following this practice>*

## Energized Work

*<provide evidence on how the team is following this practice>*