

eSpace's Software Development Process v4.0 (draft)

- [Abstract](#)
- [Introduction](#)
- [Values](#)
- [Principles](#)
- [Methodology](#)
- [Project Lifecycle](#)
- [Practices](#)
- [Quality](#)
- [Tools](#)
- [Resources](#)

Abstract

This document describes the software development process followed by eSpace to improve productivity of teams and quality of products.

Introduction

The traditional waterfall development model is a sequential process in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Analysis, Design, Implementation, Testing and Maintenance. This model originated in the manufacturing and construction industries in which after-the-fact changes are prohibitively costly, if not impossible. However, it's almost impossible for any non-trivial software project to finish a phase perfectly before moving to the next phases and learning from them.

A variety of software development lifecycles were developed over the years to better match the nature of software. In February 2001, 17 software developers met to find a common ground for lightweight development methods and published the Manifesto for Agile Software Development.

eSpace's Software Development Process is based on the values of and principles behind the Agile Manifesto.

Values

Following the Agile Manifesto, at eSpace we 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.

Principles

Following the Agile Manifesto, the following are the principles of eSpace's process:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Methodology

According to VersionOne's "State of Agile Development" survey 2011, 52% of Agile teams use Scrum. Scrum is an iterative and incremental agile software development method for managing software projects in cycles of work called Sprints. Scrum is a framework structured to support complex product development. It comes with a set of roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

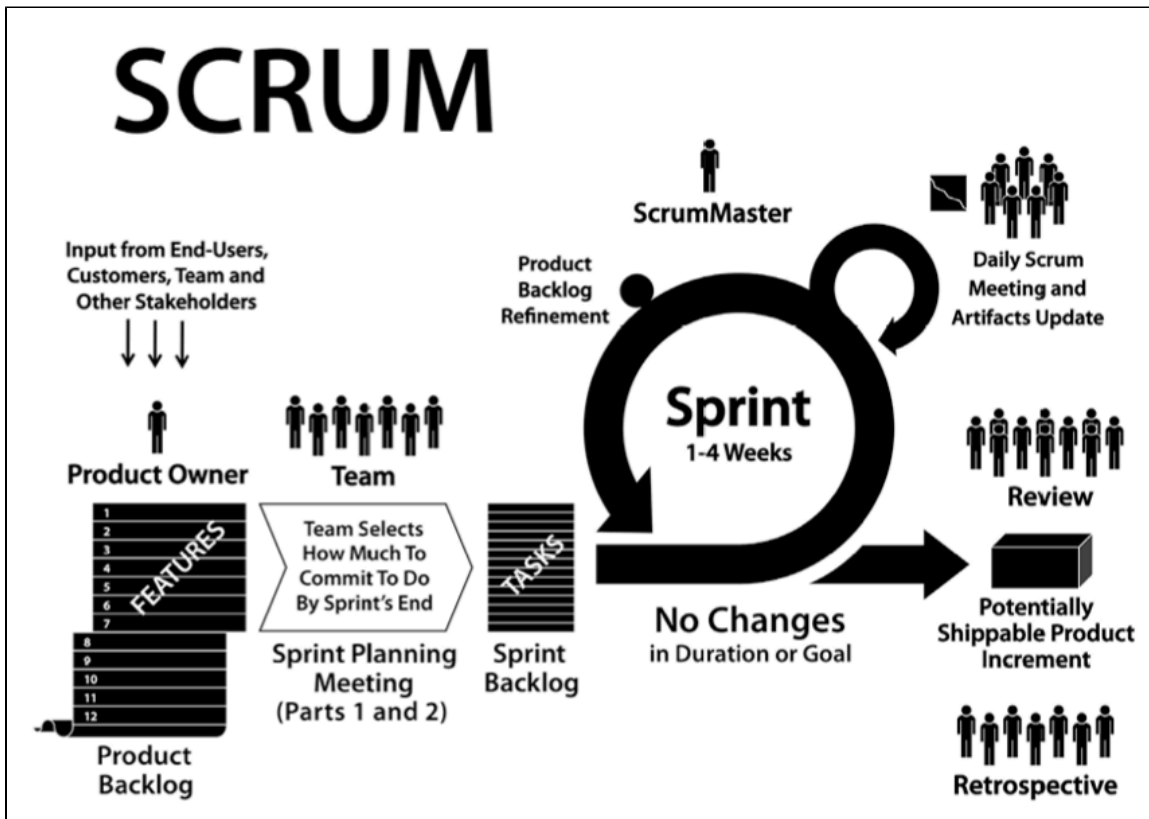


Figure 1. Scrum Framework (source: The Scrum Primer v1.2)

The current version of the Scrum Guide is attached and is part of eSpace's software development process. The Scrum Guide documents the Scrum framework and is maintained by Scrum's creators, Ken Schwaber and Jeff Sutherland.

Project Lifecycle

1. Marketing
 - The Marketing Team is responsible for mentioning eSpace's process in marketing materials such as official websites, advertising banners and project proposals.
2. Proposal [2 Days]
 - Once there's a lead, it is the responsibility of a Product Owner to work with the Client and consult a Development Team to produce:
 - i. Product vision: who the customers are, what they need, and how these needs will be met
 - ii. Product scope: Epics and Themes, what's included and what's not included
 - iii. Rough estimate of time and price
3. Initiation [1 Week]
 - a. Down payment
 - Can be X% of the rough estimate, or cost of 1 sprint depending on the project type/length/negotiation.
 - b. Contract
 - Product Owner and Scrum Master work with the Client to agree and sign a contract with the following items:
 - i. Product Vision
 - ii. Product Scope
 - iii. Definition of "Done"
 - iv. Initial Product Backlog
 - v. Initial Release Plan

- vi. Development Team
 - vii. Price and terms of payment
 - viii. Warranty
 - It should cover all the bugs report by the client. Its length depends on the project size with average of 1 sprint.
 - ix. Terms and conditions
 - x. eSpace process (Client version)
 - c. Kickoff Meeting [2 Hours]
 - The Product Owner, Scrum Master and Development Team meet with the Client to announce the official start of the project and the items of the agreement
 - 4. Implementation
 - Backlog Grooming
 - Product Owner works with the Client to order, add details and draw wireframes for Product Backlog Items (user stories or bugs) [at least once per week]
 - Product Owner consults the Development Team to provide estimates for Product Backlog Items
 - Sprints
 - Sprint [2 Weeks]
 - i. Sprint Planning [4 Hours] (Product Owner + Development Team)
 - 1. Sprint Backlog Items with more accurate estimates, in ideal days, using Scrum Poker
 - 2. Sprint Plan including implementation details and assignee of each item
 - ii. Daily Scrum [15 Minutes] (Development Team)
 - 1. What I did yesterday
 - 2. What I will do today
 - 3. What I need
 - iii. Sprint Board (Product Owner + Development Team)
 - 1. Backlog
 - 2. In Progress
 - 3. Done
 - iv. Burndown Charts (Scrum Master)
 - v. Sprint Review [1 Hour] (Product Owner + Development Team)
 - 1. What has been "Done" and what has not been "Done"
 - 2. Sprint demo
 - 3. Improvements for the next Sprint
 - vi. Sprint Retrospective [1 Hour] (Scrum Master + Product Owner + Development Team)
 - 1. What went well
 - 2. What could be improved
 - vii. Client Demo (Product Owner + Client) [1 hour]
 - viii. Documentation
 - 1. Development Team updates the Developer Guide and Admin Guide
 - 2. Product Owner updates Features and Mockups (HTML/CSS)
 - First Sprint Backlog
 - a. Setup of tools
 - b. Design of layout and theme
 - c. Investigation, if needed
 - Release Sprint Backlog
 - a. Production deployment
 - b. Performance testing
 - c. Advanced security testing
5. Closure
 - a. Customer survey and testimonials
 - b. Team closure meeting [1 Hour] (Product Owner, Scrum Master, Development Team)
 - c. Client closure meeting [1 Hour] (Product Owner, Scrum Master, Client)
 - d. Sign-Off
 - e. Celebration

Practices

The following practices are inspired by the practices of Extreme Programming:

- **INVEST User Stories:** Features are described using user stories - *As a (role) I want (something) so that (benefit).*
 - **Independent:** self-contained, in a way that there is no inherent dependency on another user story.

- **Negotiable:** up until they are part of a Sprint, stories can always be changed and rewritten.
- **Valuable:** must deliver value to the end user.
- **Estimable:** you must always be able to estimate the size of a user story.
- **Small:** possible to plan with a certain level of certainty.
- **Testable:** description must provide the necessary information to make test possible.
- **Pair Programming:** whenever possible, code is produced by two programmers on one workstation. One programmer is thinking about the coding in detail and the other is more focused on the big picture, and is continually reviewing the code that is being produced.
- **Test Driven Development:** to add a new feature, developers write a new test case, run automated tests and see if the new one fails, write some code that will cause the test to pass, then re-run automated tests and see them succeed, and clean up code as necessary.
- **Refactoring:** continuously improving the design of code by altering its internal structure without changing its external behavior.
- **Collective Ownership:** everyone is responsible for all the code and everybody is allowed to change any part of the code.
- **Continuous Integration:** frequently integrating new code to reduce rework.

Quality

- Definition of "Done"
 - Documentation updated with Features and Mockups (HTML/CSS)
 - Verified on supported browsers and devices (for example: IE 7.0+, latest Firefox, latest Chrome, latest Safari, Desktop, iPhone, Android, etc.)
 - Test coverage of 80% of code with minimum of 50% per file
 - Basic security testing
- Guidelines
 - GitHub flow: <http://scottchacon.com/2011/08/31/github-flow.html>
 - Ruby coding style guide: <https://github.com/bbatsov/ruby-style-guide>
 - Rails coding style guide: <https://github.com/bbatsov/rails-style-guide>
 - Ruby on Rails security guide: <http://guides.rubyonrails.org/security.html>
 - Google search engine optimization guide: <http://support.google.com/webmasters/bin/answer.py?hl=en&answer=35291&topic=2370419&ctx=topic>




Tools

- **Must**
 - Project management
 - JIRA with GreenHopper: <http://www.atlassian.com/software/greenhopper/overview>
 - Documentation
 - Confluence (with JIRA integration): <http://www.atlassian.com/software/confluence/overview>
 - Code hosting
 - GitHub: <https://github.com/>
- **Nice**
 - Development workflow
 - GitHub Flow: <http://scottchacon.com/2011/08/31/github-flow.html>
 - Collaboration:
 - Flowdock: <https://www.flowdock.com/>
- **Ruby on Rails**
 - **Must**
 - Unit testing
 - RSpec: <https://www.relishapp.com/rspec/rspec-rails/docs>
 - Deployment
 - Webistrano: <http://www.peritor.com/en/products/webistrano/>
 - Passenger: <http://www.modrails.com/>
 - Continuous integration
 - Jenkins: <http://jenkins-ci.org/>
 - **Nice**
 - Acceptance testing
 - Cucumber: <http://cukes.info/>
 - Integration testing
 - Capybara: <http://jnicklas.github.com/capybara/>
 - Code Metrics
 - SimpleCov: <https://github.com/colszowka/simplecov>

- Rails Best Practices: https://github.com/railsbp/rails_best_practices
- Performance testing:
 - ab: <http://httpd.apache.org/docs/2.2/programs/ab.html>
 - JMeter: <http://jmeter.apache.org/>
 - Siege: <http://www.joedog.org/siege-home/>
 - DynaTrace: <http://ajax.dynatrace.com/ajax/en/>
- Performance monitoring
 - NewRelic: <http://newrelic.com/>
- Security
 - Brakeman: <https://github.com/presidentbeef/brakeman>
 - domsnitch: <https://code.google.com/p/domsnitch/>
 - skipfish: <http://code.google.com/p/skipfish/>
 - ratproxy: <http://code.google.com/p/ratproxy/>

Resources

- Agile Manifesto: <http://agilemanifesto.org/>
- Principles behind the Agile Manifesto: <http://agilemanifesto.org/principles.html>
- State of Agile Development 2011: http://www.versionone.com/state_of_agile_development_survey/11/
- The Scrum Guide: <http://www.scrum.org/scrumguides/>
- Scrum short videos: http://www.collab.net/services/training/agile_e-learning
- Extreme Programming Practices: http://en.wikipedia.org/wiki/Extreme_programming_practices
- Software Development Using Scrum by Mike Cohn: <http://my.safaribooksonline.com/book/software-engineering-and-development/agile-development/9780321660534>

File	M
 Screen Shot 2012-03-25 at 10.29.56 ...	M
 Scrum_Guide.pdf	M
Drag and drop to upload or browse for files	
 Download All	