Software Development Methodology

Software Development Life Cycle (SDLC).

Effective August 31, 2018

Contents

1. Introduction

1.1 Purpose

This document serves as the methodology for how the development group manages and tracks software changes. It includes an overview of our software development process, as well as outlining requirements and controls associated with change management of our production software system.

This document captures core concepts and essential controls that must be maintained in our work.

1.2 Overarching Concepts

The company's core purpose is to make energy efficiency happen in the industrial sector. The software product is a tool that we build to help drive energy savings through empowering energy teams with information, direction, and motivation. In doing so, we are also mindful of company core values and our corporate commitment to doing work that is smart, sustained, and measurable. Core concepts that carry through our development philosophy include a commitment to security, a focus on the customer through functionality and delivered value, and placing value on long-term benefits for our team including maintainability and scalability.

Security

Protecting customer data is the top priority in the group. This is examined in each step of our development process.

Customer Value

- Usability
- Functionality/delivered value over form
- Availability
- Portability

Long-Term Focus

- Maintainability
- Scalability
- Extensibility

2. Product Planning Processes

2.1 Roadmap Planning

REDACTED

2.2 Design Process

Design of Features/Modules

This usually involves higher levels of feedback and input from end users, internal users, and business strategy and development.

The Process starts with a requirements-gathering process. This includes understanding the problem being solved, and the business motivation/outcomes being targeted. This leads to a design process that examines architectural requirements and defining the user experience (UX). This involves close collaboration with our customers in a feedback loop that further defines these design details.

Design of Stories

In order to move from feature design to story backlog, designs are sufficiently refined so that software development can begin. This refinement includes logical states, user interactions and responsiveness. Finally, a detailed component specification can be created. This specifies the raw data is processed, what business rules are being followed. (Business rules include formula and data processing logic for rendering data, default values). This design work involves feedback from our internal and external users, and from the development team. This iterative process is called design grooming.

Style Guide

In order to provide the best and most consistent user experience, the product conforms to a product style guide. This is our internal standard for a consistent look and behavior throughout the application.

Every story follows a design that conforms to our style guide.

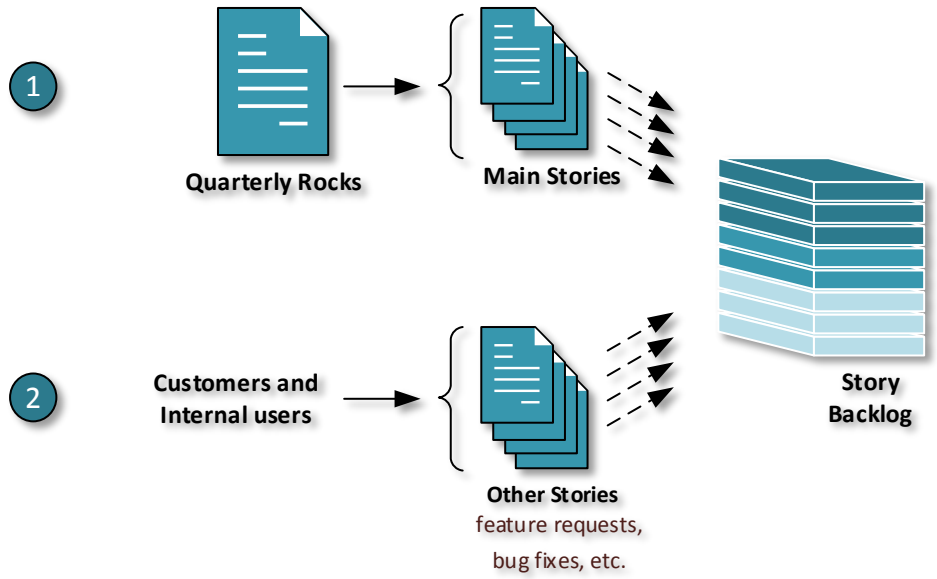
2.3 Backlog Management

The Backlog contains a list of stories generated by the product owner, customers, internal users and developers. It captures quarterly rocks as well as bugs, prioritized enhancements, and business-driven priorities. The story backlog is generally populated with all stories targeted for completion in the current or upcoming quarter and complements the longer-range feature backlog and product roadmap documentation maintained in business planning documents. All development work is pulled from the story backlog.

How Stories Enter the Backlog

- **Features** are defined in Quarterly Rocks. The Product Owner defines them into Stories and prioritizes them in the Backlog.
- **Enhancements**, lesser stories, and customer feature requests usually get into the Backlog through the IT Helpdesk, and are groomed by the Product Owner.

- **Defects** get into the backlog usually through the IT Helpdesk, and are groomed by the Product Owner.



How We Find Defects

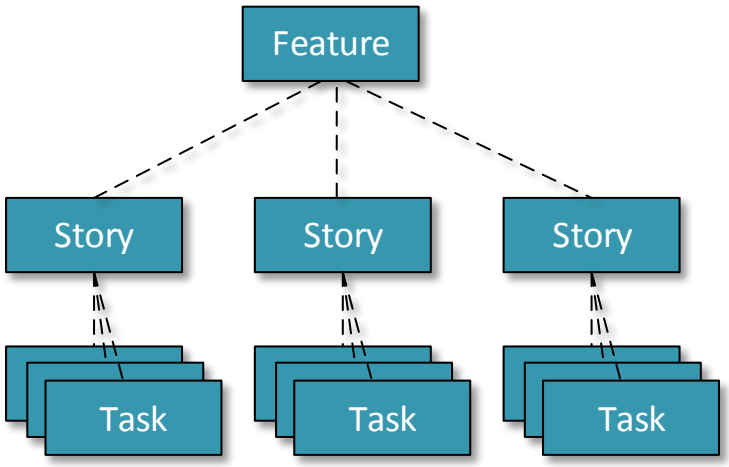
Product defects are found through these channels:

- From customers, via employees who have direct contact with them. The employees use the IT Helpdesk ticketing system for tracking.
- From developers or testers, through routine testing or observing product behavior.
- Product owner is accountable for triaging bug reports as they happen and creating appropriate stories and prioritizing

Story Backlog

The Backlog is maintained in priority order by the product owner. The Product Owner ensures that each item in the story backlog is fully specified and decomposed to a size that is completable in a single month-long sprint. Breakdown of stories into development tasks is completed as a part of the development process.

Work Hierarchy



Module	A functional grouping at top level navigation.
Feature	A fully-designed marketable, functional component. A feature could leverage or be dependent on a new microservice.
Story	A complete area of work for a functional component. A story typically is a body of work that can be completed within one sprint. For example, a defect can be resolved as a story or a task. A story consists of related tasks, all of which must be completed in order for the story to be complete. One of the tasks must include acceptance testing. A story is defined as “complete” if, and only if, it passes all acceptance tests developed for it.
Task	A single activity that can be started and completed usually in one day or less.

3. Development Process

3.1 Overview

Our Software Development Life Cycle (SDLC) is broadly based on Agile project management. This is an iterative model for program development and change management.

The development team is responsible for the committed delivery on time and on spec.

Once a software change (a feature, enhancement or defect fix) is identified as a priority, it is scheduled into a month-long development cycle called a Sprint. The development process for each work item consists of these broad categories of activity:

- Planning
- Design
- Design Grooming
- Coding
- Code Review
- Testing
- User Acceptance
- Deployment

These steps are more or less sequential, but they also overlap and are iterative to improve quality.

Each of our monthly pre-planning sessions result in a list of work items to be worked on.

Each sprint is a collection of development tasks that result in one or more completed, fully tested, and deployed features.

When a backlog item is scheduled, development on the requirements and design begins. The story begins with coding. The Requirements activity is not considered part of the story process.

3.2 Planning

The planning phase defines stories, and manages them in the Product Backlog. This includes architecture, and defining a set of tasks for each story.

This phase generates a set of technical requirements based on a story. The documentation defines the purpose, process flow, customer-facing elements (examples: screen design, functional controls, how information is presented, user assistance), data handling (including input, manipulations, transformations, validations), etc.

Activities include:

- Specify grouping of new features, improvements, bug fixes, scoping.
- Specifies the amount of effort required, when and how it is released (e.g., coordinated with other new features)
- Staging method (could identify beta testers)
- Determine acceptance criteria
- Determine deployment date
- All requirements are clear, precise, detailed, realistic, not contradictory, aligns with the vision of the project, agreed upon by stakeholders.
- Optional: develop testing model

3.3 Design

The Requirements are translated into an initial technical design specification.

The implementation is indicated in the design. No “leap of faith” considerations are implied in order to code. The design conforms to our design standards, and ensures that there are no issues that may impede the Coding phase.

Design Activities include:

- Design aligns with the architecture
- Requirements can be revisited if necessary

3.4 Design Review (Grooming)

Design Review is an iterative step with Design to ensure that all requirements are met in the design specification. This is led by the Product Owner, and involves one or more designers and an architect.

3.5 Coding

This is where the change is built. It’s part of an iterative process based on feedback from Code Review, Test, and Verify.

All code meets our internal coding requirements for our “-ilities”, such as accuracy, security, maintainability, consistency, and our overall design standards prior to Code Review.

3.6 Code Review

Code Review is an iterative process with the Coding phase. The completed code is peer-reviewed by two other developers to ensure that it meets the design specification, our coding standards, and that no logic errors or implementation issues are found. Code review includes coding best practices and security issues.

All new and modified code is peer-reviewed by two other developers to ensure that it meets the design specification, our coding standards, and that no logic errors or implementation issues are found.

3.7 Testing

We ensure feature quality through acceptance testing. Testing often reveals coding defects, and becomes iterative with coding.

All new and modified code is quality tested to ensure that it is reasonably free from defects, and will not impede normal operations in the production environment. Customer acceptance testing occurs here.

3.8 Deployment

Deployment is allowed when staging has been tested and approved.

Ensure that all new and revised code has passed user acceptance testing.

Deployment Activities include:

- Review for production readiness

- Release to Production
- Verification testing. Our internal product users perform continuous live usage to ensure a stable and quality product.
- Defect handling. When issues are discovered, the product owner alerts to the development team. The development team works with the product owner to determine the most appropriate resolution, and get it deployed in a timely manner. The SDLC model is used to track the issue and resolution from design to deployment.
- Customer communication - Internal customer feedback is essential during deployment. This is our primary feedback mechanism and representation.
- Documentation

Customer communication leads to:

- Defect identification and quick fixes
- New feature requests from outside
- System maintenance and enhancements from the application itself

3.9 How Emergency Fixes are Handled

Occasionally, very high-priority work items like custom patches, emergency fixes, and roll-backs are handled outside of the normal monthly commitment process.

When an issue requires quick action, the product owner is consulted to evaluate the impact, possible solutions, developers consulted to identify possible fixes, product owner decides on the fix/solution. Depending on the urgency, alternate testing methods and procedures may be employed.

In all cases, changes in deployed code must be reviewed by two or more developers, and must be tested before deployment to ensure that it resolves the issue and does not introduce new issues, documented in the Story Board, and then reviewed in subsequent planning sessions.

4. Appendix

4.1 Glossary

Term	Definition
Done (Definition of Done, DoD)	DoD is a shared understanding of expectations that software must live up to, to be releasable into production. This is managed by the Development Team. It is used to determine if an activity in the Sprint Backlog has been completed. DoD is a quality assurance measure. DoD varies according to context: • DoD for a Scrum Product Backlog item (e.g. writing code, tests and all necessary documentation) • DoD for a sprint (e.g. install demo system for review) • DoD for a release (e.g. writing release notes) This includes compromises if late and need to complete w/ redefined Minimum Viable Product.
Rock	Rocks are the strategic priorities and action items that govern where each team member puts their primary effort. They are created or reviewed quarterly, and are characterized as: specific, measurable, attainable, realistic and timely.

4.2 External References

- https://en.wikipedia.org/wiki/Scrum_(software_development)
- https://www.versionone.com/agile-101
- "Traction: Get a Grip on Your Business," by Gino Wickman. Discusses the concept of quarterly company Rocks.

4.3 Version History

Version	Date	Author	Changes Made	Approval
0.5	8/31/2018	L. Humbird	First Edition	n/a