RelayHealth

VERSIONONE Quick Reference Guide

VersionOne Enterprise XP Fall '12 (12.3.6.74)

Version: Draft 4

Version History

Version	Document Author	Revision Date	Approved By	Approval Date	Reason
Draft 1	Gay Stahr Paul Dobson Sean Kiner				Initial Draft
Draft 2	Gay Stahr Paul Dobson Sean Kiner	4/8/2013			Additional content after CMMI group meeting.
Draft 2	Gay Stahr Paul Dobson Sean Kiner	4/8/2013			Added attributes for defects
Draft 3	Paul Dobson	4/102013			Reword and Defect Attributes sections
Draft 3	Gay Stahr	4/18/2013			Edits
Draft 4	Paul Dobson	4/25/2013			
1.0	Gay Stahr Paul Dobson Sean Kiner	TBD			Final Edits and review

Table of Contents

USING VERSIONONE

OVERVIEW

Big Picture

Source: VersionOne Enterprise User Guide

In Product Planning the team composes, estimates, and prioritizes Backlog Items. Theme groups, epics, goals, requests, and issues can also be created to assist with managing projects.

Once a product backlog exists, the Product Owner has the option of organizing these backlog items into one or more releases in Release Planning. Teams can also be set up to divide work among functional groups of team members.

Within a release, backlog items are further organized into iterations.

During Iteration Planning the team defines the acceptance tests and identifies all tasks for each backlog item in the iteration.

Once a story is assigned to an iteration, the 3 amigos (resource roles) are assigned to it (development, QA, and BA). This team is responsible for making sure the story is complete (including acceptance criteria, testable, and is complete after QA has approved it.

During the iteration, team member use Iteration Tracking to update progress and close backlog items once they have been accepted by the Product Owner.

At the end of the iteration, the team uses Review to close the iteration and conducts a retrospective. The process repeats for the next iteration.

STANDARDIZATIONS

Using Themes

Themes are not intended to be identical across projects or teams. They are a method of categorizing types of work in a way which makes reporting from VersionOne more convenient. Therefore, they shouldn't replicate other fields such as priority and type. And they shouldn't be used in the manner intended for epics.

For instance, a theme called "Critical Defect" is redundant, as is a theme called "Tech Debt", since other fields can be used to provide these attributes. If you have a number of stories with a theme that mimics something like "UI - CMS Form 1500", you are probably using a theme where an epic should be used. A theme is best used to describe component level functionality, or as a replacement for a field that doesn't exist (although custom fields can be used for this).

Good examples of themes might be something like the following:

Themes – Functionality

- Administration
- Customer Facing
- Reporting
- Environmental

Themes – Target Deployment / Environment (could be replace by customer fields)

- Production
- Training
- Support Tools
- Collaboration Compass

Exceptions:
There are rare occasions where a general theme may help organize activity across multiple projects, in a way that attempting to use an epic could become a source of confusion. For instance, ICD-10 might fit this description. In this case, a compliance mandate that has broad implications presents a singular issue for the organization.

Reference material:

- http://community.versionone.com/KnowledgeBase/FAQs/Q11168.aspx
- http://community.versionone.com/Lists/Announcements/DispForm.aspx?ID=53

Using Epics

An Epic is a large-grained feature that is broken down into smaller components for individual estimation and scheduling. This cycle may occur multiple times as these components can be broken down into even smaller components. Use epics to capture and define big features that are eventually decomposed as a natural part of the planning process. Epics are most appropriate for those items that you might want to schedule as an individual feature, but end up being too large for planning purposes. A workitem can belong to only one Epic, but can be in any Project in the hierarchy. Epics can be organized into a hierarchy of multiple levels, and can be composed of a mix of Stories and Defects.
Examples of Epics:

- product search
    - simple search
    - advanced search
        - search by location
        - search by fuzzy name match
        - search by multiple fields
- warehouse system integration
    - inventory management integration
    - shipping integration
    - receiving integration

The key difference between an Epic and a Theme is that you can put a status on an Epic, much like you can have a status on a workitem. Also, Epics are more of a finite amount of work, while Themes are more enduring classifications of functionality.

Rework

Iteration Rework

For the purposes of VersionOne, the Rework status is used when a story must reenter the coding stage due to a defect or a change in story content. Stories should not be converted to defect work item types (from S- to D- work items) unless they began the iteration that way.

The intent of the Rework status is to identify patterns illustrating chronic patterns by individuals or component work that should be addressed.

Stories & Defects

Defects that are found during regression testing or escalated by Customer Support to Development, are a type of rework, but are not assigned the Rework status in VersionOne. If a defect is technical in nature (that is, it's not related to ambiguous requirements), it's created as a Defect work item (D-nnnnn) and entered in the backlog (or assigned to an iteration).

If a story is discovered to be the product of an ambiguous or incomplete requirement after the iteration has ended, the following example scenarios can occur.

- The code as developed isn't acceptable for deployment, the story hasn't been closed and it hasn't been deployed to production: The story is moved into the backlog and corrected. The original code is removed from the source code and a new build is distributed to QA.
- The code as developed isn't acceptable for deployment, the story has been closed and it hasn't been deployed to production: A new story is created and entered into the backlog. The original code is removed from the source code and a new build is distributed to QA.

- The code as developed provides unintended functionality, but can be corrected later. The story is closed and a defect is entered into the backlog.
- The code is incomplete, and can be updated later. The story is closed and a new story is entered into the backlog.

Implementation

The implementation of the attributes defined in this document will be completed as follows:

- All new options for the above attributes will be added prior to the 1st occurrence of FY Iteration 3 on April 29, 2014
- Only the documented options will be available upon start of the 1st occurrence of FY Iteration 5 on May 20, 2014. That is, all options not documented will be removed
- The lists above will be sorted in the order given
- If an "Unplanned" custom checkbox field is created, it's default value will be unchecked

VersionOne Attributes

Story Attributes

Done: The Scrum Team is responsible for shared ownership of *Done*

- code complete and unit tested
- acceptance tests executed and passed
- agreement by all 3 Amigos that story/defect meets requirements

Priority: During or prior to Release Planning, the Product Owner assigns a priority of every story (High, Medium, Low) so their order can be determined and rough estimates made. During iteration planning, stories assigned to the iteration are assigned either Commit or Stretch priorities, to indicate their planned outcomes:

High .....High business priority – set during Release Planning

Medium .....Medium business priority – set during Release Planning

Low.....Low business priority – set during Release Planning

Stretch .....Worked during the iteration, if there are resources and time to complete the story – set during Iteration Planning

Commit.....Must be completed during the assigned iteration – set during Iteration Planning

Type: A classification grouping specific to an asset type (referred to as category in prior releases)

New functionality .....Indicates either net new features or enhancements

Production Issue .....A defect or other production issue that is worked during the iteration

Tech Debt.....Coding framework or setup, database DDL, environmental setup, etc.

Unplanned .....Any story not planned for during Iteration Planning

Source: Indicates the source of the story and its priority

Product Management...Stories authored by, or approved by Product Management - even if other teams requested them (Registration, Implementation, etc.)

Support .....If the source of the story is a production issue (i.e. a service order)

Development.....The story is tech debt, or a recognized need to resolve a technical issue

Defect Attributes

Defects are typically created outside the iteration. That is, when a production issue is discovered and must be addressed by development, or during regression testing. It's also possible that a misunderstood (or mis-communicated) requirement caught late must reenter development. In these cases, the defect is assigned to the backlog, or immediately to an iteration.

Priority: A defect can be assigned a priority at anytime – prior to or during an iteration. Priority is the outcome of an agreement by the product owner, QA analyst (if found in the development cycle) and Development Manager or Scrum Master. The Defect Priorities in VersionOne are aligned with the Severity code in ALM.

The table below reflects the meaning of the values for both.

Priority / Severity	Description
Critical	Defects that renders the system unable to continue normal application functions of the business process and are critical enough to include the following: • System crashes, hangs, locks up or ceases to operate • Causes loss of data or file system corruption • No work around to the problem exists
High	A defect that renders the system unable to perform normal operation in areas of high priority in the business process. These errors are high enough to include the following: • System still up and running but cannot access data • Some feature is non-functional • Costly or difficult work-around • Defects are in non business-critical modules or functionality • Inability to print forms or generate audit reports etc.
Medium	Defects of non-critical features for which a convenient or reasonable work-round exists. Defects that have a medium effect on the business process may include the following: • Performance issues or "set-up" issues. • Usefulness of some feature is impaired, easy workaround exists • Problems accessing printers or printing other than business forms • Database errors or errors that arise while viewing non-critical reports

Priority / Severity	Description
Low	Defects that impact upon the business process at a low level of performance, or where a request for information is not functioning that does not impact upon ability to continue normal day-to-day business operations. Defects in this category may include the following: • Problems accessing the help file or help functions • Problems with "drill-down" activities or reviewing non-essential history • Cosmetic problems with the interface or menus, misspellings or inconsistent color schemes • Problems with issues where a work-around is available and are minor in functionality

Type: A defect classification

Development..... Defect originated in design, database, code

Config / Environment... Issue due to the environment itself or its configuration

Data..... Issue with data (database or file)

Requirements..... Omission, ambiguity, or change from what was agreed upon

3rd Party..... Due to software or system external to the product or its environment

Other..... Not categorized (nature of defect should be reflected in defect description

Source: Indicates the source of the defect's discovery

QA..... Defect found during development cycle – during iteration or regression testing (defect can be discovered in cycles after the defect was introduced)

Support..... If the source of the story is a production issue (i.e. a service order)

Development..... Found during development or unit testing, and required a separate defect to be spun off and put in the backlog

Other..... Defect discovered by personnel outside the above roles

Best Practices

Tracking Effort

Story level estimates should only be entered into the **Estimates** field. These may be in the form of story points or hours, but must be consistent across the project.

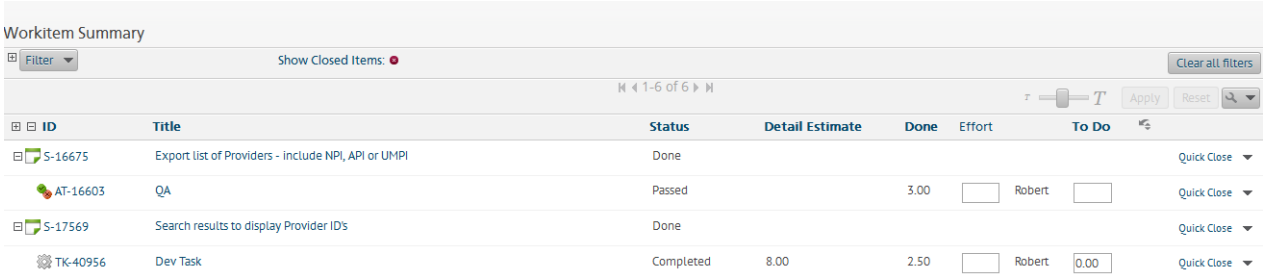
Detailed estimates are entered at the *task* or *test* level in the **Detailed Estimate** field. This field is rolled up to a display field called **Detailed Estimate** at both the story and iteration level.

To enter and track effort, and assure a correct iteration burn-down:

- Enter hours in the **Effort** field

- Decrease the hours in the **To Do** field (or increase if estimated work has changed)
- Save the updates (either the **Apply** button in the *Detailed Tracking* screen, or **OK** in the *Task* or *Test* dialog
- When the *task* or *test* is saved, any **Effort** added will increase the hours displayed in the **Done** field
- The **Estimate** field should never by updated once work on the story has begun

My Home – My Work View



Closing Stories

- Stories can be closed as soon as they enter the **Done** status
- If a story is scheduled for production deployment during the iteration, the story should go to a **Deployed to Production** status before it's closed
- If a story is **Canceled**, it should be closed immediately after entering that status
- In all cases, stories should be closed and all **To Do** hours should be set to 0 before the iteration is closed

Tracking Iterations

Iteration Summary

Total Estimate: A total of all of the Story and Defect size (in **Estimate**) for the iteration.

Detail Estimate: A total of all of the task and test **Detailed Estimate** hours for the iteration.

Total Done: A total of all of the **Effort** entered in this iteration. If effort was recorded for any of the tasks or tests outside of this iteration it will not be included in this total.

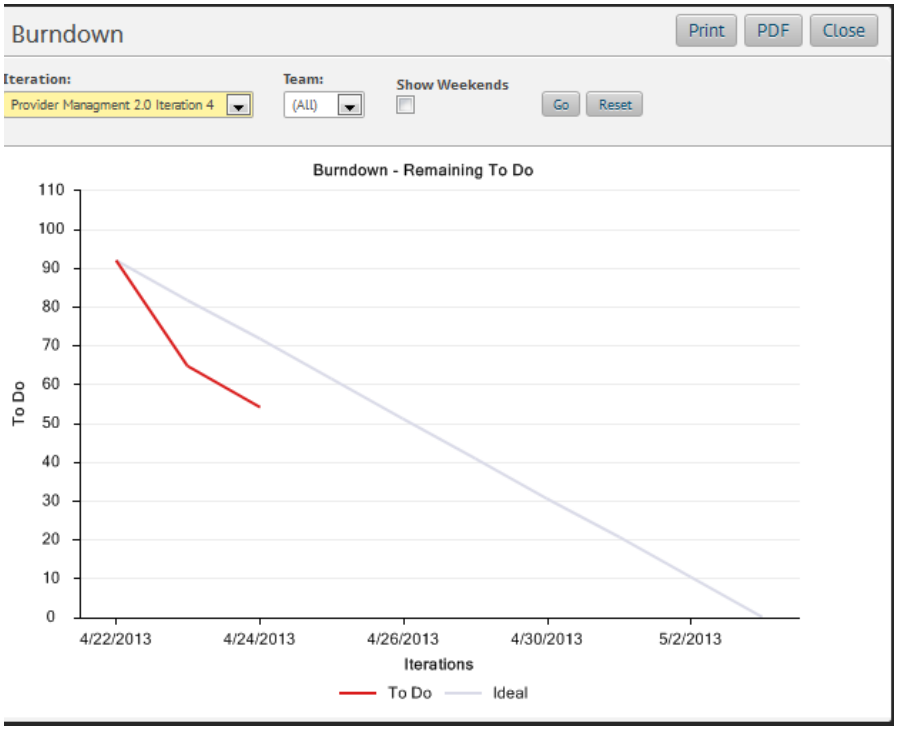
Total To Do: A total of all of the remaining effort **To Do** in this iteration

Iteration Burn-down

The graph shows the following lines for the selected iteration:

To Do: Shows the sum of all **To Do** values in the iteration as of the end of the day (red Line).

Ideal: Shows the ideal slope of the Burn-down using the highest **To Do** amount for the iteration as the starting point on day one.



Release Planning

Total Estimate: Represents the totals of the Estimate field for all stories, including closed and stories assigned to open, closed, and future iterations, and those stories in the project backlog

Total Estimate - Rollup: Represents the above estimate as described above, plus those of all child projects.

UPDATING VERSIONONE

VersionOne (V1) is primarily an Agile management tool used to manage the software backlog and the software creation lifecycle. For distributed team members and reporting purposes, the backlog is a critical medium to maintain up-to-date progress status.

- V1 is the source of truth
- Clearly watermark all external documentation that occurred prior to story-load into V1 with "**Superseded by V1 xx/yy/zz**"
- An external overview/set of flows is acceptable but must be clearly marked as supplemental material

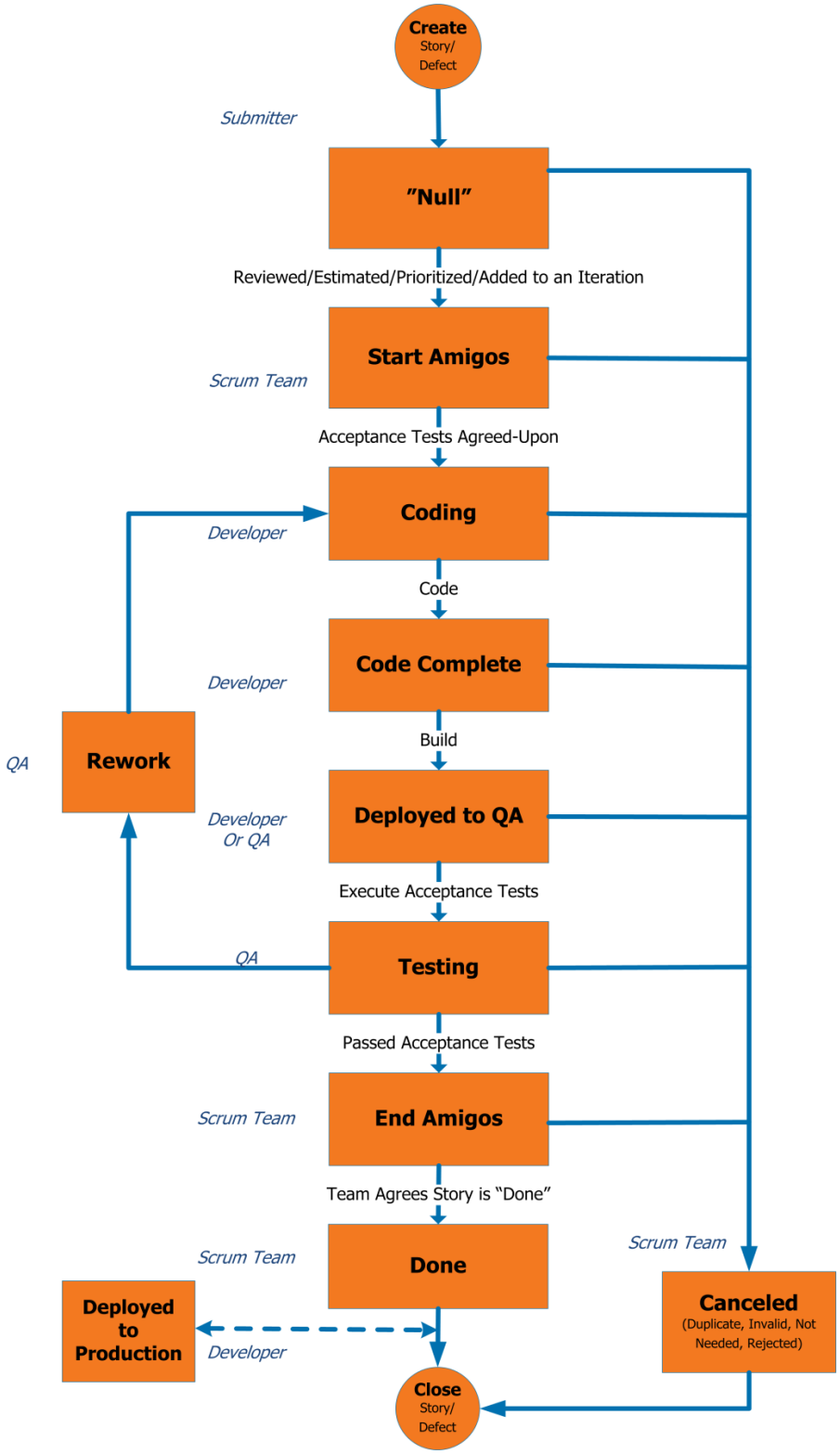
V1 Responsibilities

Stage	Responsible	Action
Themes	Product Owner	Enters the primary features to be included in a release
Epics	Product Owner	Enters a high-level story that describes the functionality from the end user's point-of-view
Stories	Product Owner	Enters a story that describes the functionality from the end user's point-of-view: As a…I want…So that
Projects	Project Manager	Enters under parent product and assigns project members
Release Planning	Project Manager	Update V1 to match commitment
Iteration Planning	Scrum Master	Update V1 to reflect Iteration commitment – assigns stories to Iteration
Iteration Schedule	Project Manager	Adds Iteration schedule
Daily Stand-up	Scrum Team	Update V1 prior to meetings
Iteration Review	Scrum master	Close out Iteration in V1
Iteration Retrospective	Scrum Master	Update V1 to reflect results
Backlog Priorities	Product Owner	Sets priorities
Backlog Grooming	Scrum Master	Ensures that grooming is an ongoing activity
Backlog Evolution	Scrum Team	Participates as a partner in the real-time evolution of the backlog
Burndown	Scum Masters	Shows burndown at every stand-up
Reports	Scrum Team	As required by Software Creation Process and/or by management

V1 Iteration Story Status Codes

Responsible	V1 Status Code
Scrum Master	None — when first assigned to Iteration
Amigo	Start Amigos
Developer	Coding
	code Complete
Scrum Team Member	Deployed to QA
Tester	Testing
	Rework
Developer	– Coding
Scrum Team Member	– Deployed to QA
Tester	– Testing
Amigo	End Amigos
BA/SME	Done

V1 Status Flow Diagram



RESOURCES

VERSIONONE RESOURCES

Requesting Additional Licenses

Contact the director, or a manager within the RelayHealth Financial PMO.

System Administrators

Current system administrators include:

- Paul Dobson
- Julie Poole
- Sally Mihalakis

VersionOne Support Center

VersionOne online support offers extensive instructions in convenient online formats that include simple explanations, discussion forums, videos, and a knowledge base. All of these formats are easily accessible and usable during operation of the V1 software. The online Support Center offers help in the areas of:

- Getting Started (login required)
  http://dbqver2.tsh.mis.mckesson.com/VersionOne/Default.aspx?menu=MyHomeEnterpriseGettingStartedPage
- Help (login not required)
  http://dbqver2.tsh.mis.mckesson.com/VersionOne/Help/XP/V1Help.htm
- Support (login not required)
  http://community.versionone.com/default.aspx