# Yoda - All-In-One Project Management with GitHub

Jens Vedel Markussen

HPE, Communications and Media Solutions

jens.markussen@hpe.com

## Abstract

*We have developed and released into Open Source a light-weight - yet full-fledged – Agile Project Management tool, code named Yoda. To avoid the tool-disconnect often seen between developers and project managers, Yoda has been done as an extension to GitHub. Defects and enhancement requests are fully documented as GitHub issues, which are collaboratively maintained by Product Managers, Project Managers, Developers and Testers. In short, Yoda has enabled our product Organization to handle source code, defects, customer discussions, prioritization, estimation, back logs, sprint planning and tracking, reporting, dash-boarding and release notes generation in one single solution – GitHub/Yoda.*

## Problem statement

GitHub is a popular software collaboration system combining source code management with issues management. HPE's in-house deployment of GitHub is the system of choice for managing code and issues. While GitHub can associate milestones with issues, it does not support estimating, tracking and reporting work associated with such issues. Project management should not be handled in a separate system as this leads to information duplication, synchronization problems and hampers alignment between project managers and developers. Finally, we want to remove communications barriers between the development organization (R&D) and our clients.

## Our solution

We have augmented GitHub in order to provide a single system for supporting all project management related activities in addition to its core task of keeping track of source code and issues. Our solution combines project management best practices with the *Scaled Agile Framework (SAFe*) methodology. It consists of these parts:

1. Conventions defining how to manage sprint-based work items using GitHub issues and milestones.
2. Process descriptions covering main product process flows using conventions from 1.
3. A browser-based GitHub extension (Yoda) for project management based on and supporting 1 and 2. All functionality is based on data stored in GitHub.

### GitHub Conventions

Fields in GitHub are restricted to the bare minimum. For issues, this includes mainly the issue title and description, the assignee(s), a link to the associated milestone, and a generic label mechanism. There are no explicit fields to denote issue type, the severity of a defect, or the estimated effort associated with the issue. To capture this additional information, we turn to conventions for *labels* and for the *description text* of the GitHub issue.

GitHub *labels* are really just text strings with an associated color, e.g. red background for a critical bug. Multiple labels can be associated with an issue and are maintained per repository. We have put in place an *issue labelling convention* where labels are grouped as indicated by a common prefix and color. The most important classification is the issue *type*, prefixed by a `T`. The options are `T1-Defect`, `T2-Enhancement` and `T3-Task`. Other important label classifications are *severity* (`S-`), affected *customer* (`Co-`) and *process* (`P-`).

Some data types do not lend themselves well to labels. Instead, we keep such data in the GitHub issue *description text* using a `> field-name field-value` convention. This easy notation also renders nicely when GitHub interprets it as Markdown text. We use such fields for issue estimates, remaining work and release note details.

Inspired by SAFe we use *story points* as the unit of estimation. Story points can be a somewhat fluffy concept, so we tend to think about more tangible *man days* during estimation. It is important to not link story points directly with available man days. Rather, retrospective *velocity reports* are used to work out a story point to man day factor, which is unique per development team. This factor is taken into account when planning subsequent sprints.

In modern agile planning - including SAFe - the mantra is to break-down all work into small chunks of only a few story points. This avoids the need to keep detailed track of remaining work; if the issue is open, the full estimated work remains, once closed, no work remains. In reality it is difficult to always do this breakdown, so we allow developers to annotate remaining work for bigger items. This enables Yoda to build an accurate *burndown* chart.

GitHub *milestones* are used to denote product *sprints* with associated issues indicating when the associated work is planned to complete. GitHub milestones only have a title and a due date, so we augment this with a start date, an optional development completion (code freeze) date, and a value for the team capacity in story points.

By following these conventions, GitHub issues provide an accurate view of all sprint work and status.

## Processes

We have described and implemented processes covering aspects of *product management*, *product development* and *testing*, and *support*. As GitHub does not include a workflow or state engine, the processes can be seen as *soft processes* by convention. Well-defined and -understood processes are critical to the success of our programme.

Processes rely on GitHub issues for discussions and sharing of ideas. Labels-, milestone- and people- assignments guide processes along. We use markdown formatting and image embedding. GitHub repositories have full read access, with full transparency to all interested parties, as opposed to a classic “walled garden” R&D approach.

**Project Communication Process** addresses communication between customer project teams and the R&D team. We assist new projects by creating a project specific GitHub repository, including labelling conventions as per above, where the project raises issues (defects or enhancement requests). If a product defect is identified, a corresponding product issue is created in the product repository and the two issues are bi-directionally linked by GitHub references. Enhancement requests are handled in the same way and created as product issues if Product Management accepts them. Regular R&D/customer project calls review the issues in the project repository.

**Prioritization and Sprint Planning Process** assigns priority and plans issues representing Product Enhancement requests from customer projects or Product Management. Labels indicate which customer(s) have raised - or have interest - in the enhancement. Based on known issues product Project Managers provide a high-level estimate for the enhancement. This list is used during monthly scoping sessions, deciding which enhancements will be addressed in the upcoming monthly release sprint. The milestone of each issue is set to indicate the appropriate sprint. Enhancements may be accepted as *tentative*/non-committed as indicated by a `P-Tentative` label.

**Support Process** ensures that customer support cases received via the HPE support system, *SalesForce*, may be elevated to R&D/Level 3 in case the support engineers doing level 1 and 2 trouble-shooting need to do this. If so, they will raise a GitHub issue of type `T1-Defect` with a special `Support` label and assign it to the R&D Release Manager, who will assign it to the proper developer or tester for further investigation. All communication between R&D/level 3 and the Support organization is recorded into the GitHub issue, leaving a full audit trail. If the support case requires a product fix, it is assigned a milestone - typically matching the next sprint release.

**Progress Tracking** is done based on up-to-date issue data provided by developers and testers on a regular basis. This includes closing issues when complete or updating the remaining number of story points. This process allows up-to-date views on sprint progress using Yoda, at component or at product level.

## Yoda

Yoda consists of a set of tightly coupled tools running in a browser. Yoda is served from GitHub’s built-in web server (GitHub Pages) voiding the need for any product installation. All data is kept in GitHub without any auxiliary database. Yoda tools may be run interactively in a browser or embedded into e.g. live *dashboards*. Tools may be categorized into *Planning and Tracking, Reporting* and *Maintenance Tools.*

The primary *Planning and Tracking* tool is the sprint *burndown chart* which looks at sprint efforts across one or more subcomponents (represented by issues in component repositories), (see Figure 1). Estimates and remaining values are extracted from issues, and plotted against sprint dates defined on the milestone. A solid line denotes the *ideal burndown* towards the sprint end date. Issues marked as *tentative* are shown in yellow.

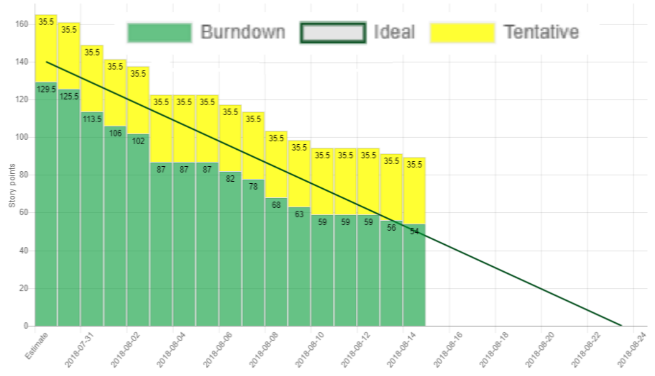

*Figure 1: Yoda Burndown Chart*

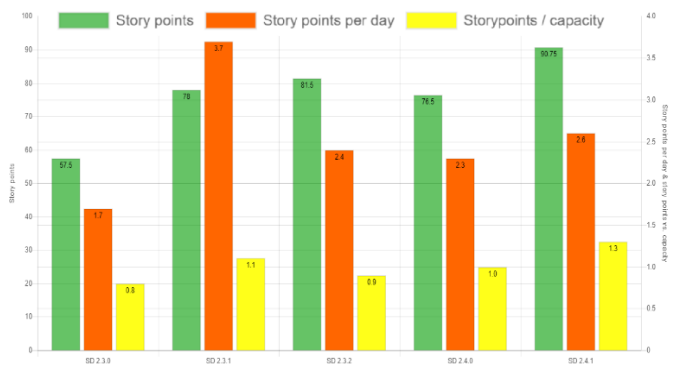

*Figure 2: Yoda Velocity Report*

To help calibration of story points to man days, Yoda supports *velocity charts* (see Figure 2) which look at completed story points across sprints. Absolute story points accomplished are reported alongside normalized values (in case of diverse sprint durations), as well as a comparison between the story points done and the set capacity. Ideally, this value should converge towards one, as planning accuracy improves.

Yoda *Reporting* shows the evolution of issues over time, either as simple bar charts or *Cumulative Flow Diagrams (CFD)* showing both open and closed issues over time. CFD charts scoped at defects or enhancements provide an at-a-glance overview for the Quality Manager or Product Manager, respectively.

Yoda has *Maintenance Tools*, e.g. to ensure label and milestone consistency across repositories. An aligned repository for a new component or customer delivery project can be set up in minutes. Yoda can generate release notes based on issue data, including detailed descriptions written directly by the developers - giving a greater sence of responsibility to the developer, than if such tasks are left to Release Managers or Technical Writers.

## Evidence the solution works

Our solution has enabled requests to be prioritized, developed, tested and delivered within 4-6 weeks as opposed to classic waterfall times of 3-6 months. Project managers and developers get up to date information on sprint tasks and status, including indications of any potential date slippage risk. Management can interactively access the live dashboard at their convenience. Communications between projects and R&D is smooth with a full audit trail, and both projects and customers are providing positive feedback about the transparency provided by the solution.

## Competitive approaches

Various commercial products target GitHub deficiencies in the project management space, including ZenHub and Zube. At 5-10 USD/month/seat both deliver fewer features than our solution by *replacing* rather than *complementing* the GitHub user experience. They also replicate GitHub data to their own databases, adding complexity. Another popular collaboration platform, GitLab, has recently included basic project management features. While still immature, it shows that including project management into source code/collaboration tools is a sound idea. Jira is a project management, issue- and requirements- tracking tool (also deployed at HPE), which includes a process/workflow engine. However, Jira does not provide integrated source code management.

## Current status & Next steps

As of today (November 2018) GitHub/Yoda has been in operation for about 12 months. A significant number of developers, testers, product managers and architects in the HPE ecosystem use GitHub/Yoda every day to manage underlying repositories with more than 15,000 issues.

Yoda was released into OpenSource using an MIT license in January of 2018 and is continuously evolving based on user input coming both from HPE internally as well as from the greater OpenSource community.

## References

Yoda is fully described via the Yoda landing page. Information on SAFe can be found here. Yoda source code is available in this GitHub repository.