Tools

Notes on tools organized by tool domain.

Software testing

Robot Framework

Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has easy-to-use tabular test data syntax and it utilizes the keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new higher-level keywords from existing ones using the same syntax that is used for creating test cases.

CircleCI

Let CircleCI help your team focus on making a great product. Speed up your testing and development cycle to improve productivity. CircleCI is flexible to run in your environment and scale with your growth. Have the peace of mind by reducing bugs and improving the quality of your application.

- Fast, automatic parallelism and intelligent test splitting
- Quick & Easy Setup
- View build status from GitHub or see PRs and commit messages from CircleCI.
- Beautiful User Experience
- Real-time Build Output and Detailed Test Failure Data
- Smart Notifications
- Inferred Test Commands and Pre-installed Packages and Services
- SSH Access
- Start free and scale without limit
- Free for OSS
- YAML-based Config
- RESTful API and Webhooks
- Build Artifacts
- Sudo Support
- iOS and Android
- Docker
- Deployment Keys and Secrets

Backlog management

General

- Sprint template
- How to maintain a lean product backlog

Agilefant

Agilefant is an end-to-end project management tool that supports also product planning and portfolio management.

- it is a hierarchical backlog tool: *Product* > *Project* > *Iteration*, and *Story* > *Task*
- *iteration burndown* is a graphical representation of tasks' progress in the iteration as a function of time: It shows a linear reference based on the sum of the tasks' *original estimates* (stories must be split into estimated tasks).
- stories without child stories are called *leaf stories*
- stories can be anything from business goals, customer requests and features to small issue descriptions
- a story consists of zero or more tasks but tasks can also be unlinked to any story
- tasks are used to describe the work needed to get the story done or some small tasks such as meeting or phone call
- a story must always reside in a product, project or an iteration
- stories in the product backlog have no particular *priority*
- a story can then be assigned to a project and/or an iteration where it can be prioritized seperately for both
- story states:
  - *Done* – it reflects the DoD and is thus considered in relevant metrics
  - *Deferred* – the story has been decided to be skipped in this iteration; the *effort left* / *points* are omitted in all metrics; another option is to move the story to another project/iteration
  - *Not Started* – No work has yet been put into realizing this story
  - *In Progress* – ongoing and some work has already been put in
  - *Pending* – waiting for something external that can reasonably be expected to happen without us taking any further action
  - *Blocked* – can't proceed; most likely some action must be taken by 'us' before work can proceed
  - *Ready* – done if only some relatively minor DoD criteria gets met: peer review
- stories may also be assigned a *business value* to facilitate sprint planning

- stories can be *labeled*, f. ex: bugs, usability improvements, strategic functionality, and so on
- stories can be set to depend on other stories
- stories can have attachments (images, files)
- a task is something relatively well-defined and effort-wise small that needs to get done
- tasks are estimated in man-hours; *effort left* refers to the hours left to get the task done. A story's *effort left* is the sum from its tasks. A task's first effort estimate is its *original estimate* which affects the iteration burndown's estimate line.
- tasks also have *spent effort*, the effort already worked on it
- if you add spent effort entries within 8h of each other, the tiem difference is given as default when logging the next entry
- users can be set as responsible for stories and tasks to indicate who is working on what, note that task responsibilities are not propagated upwards to stories

Resources

- FAQ
- User guide
- Introduction and usage example

Trello

- Scrum for Trello
- 10 tips for using Trello for Scrum
- 5 tips for using Trello for Scrum
- Trello-Flowdock integration

Collaborative editing

Floobits & Sublime

- Provides realtime simultaneous file editing collaboration cababilities.
- Can be setup with a repo such that changes are easy to reflect back in GitHub.

See http://awan1.github.io/subl-floo-tutorial.html