Plan

1. research existing plans -- extract patterns -- what problems do they solve, what are the classes of patterns of solutions
2. what problems is Drupal.org trying to solve
3. what are the patterns used now on d.o? What anti-patterns does it follow?
4. What are some possible changes? How to transition? What's the low-hanging fruit? What parts are unchangeable for now for technical/social/other reasons?
5. Post on d.o. for review
6. prototype new design(s) in html/js

Screenshots:
http://www.flickr.com/photos/82268668@N00/sets/72157627969963444/with/6278075102/

Previous Drupal discussions:

- http://groups.drupal.org/node/144574 - this is from the prairie initiative and links to most other relevant discussions

What is an issue queue?
how we do work = define our task then do it

Issue queue helps define what the task is then ensures that completed work fits what was intended.

It does this by bringing together all the needed information (feature x is broken! I need help on y! This issue is dependent on z!) and the right people (holders of tacit knowledge) in one place so that the task can be defined correctly, assigned out, and monitored until it is finished.

So three stages. Identify work, bring together information and define work, monitor progress of work until finished.

But there isn't strong distinction between stages. All are continually operating on each other. Doing work helps us better define what the task is which changes what work needs to be done. Each issue must help people working continually update issue so that the definition of the task is always current and the progress of the work is easily monitored.

Information + people to help define the work. Up-to-date definition of work. State of progress toward that goal (i.e. who's doing what, what have they done, and what's left to do).

Github's pull request tool does a brilliant job of pulling together all these elements.

Ways to improve issue queue.

1. Make it easier to bring together the right information/people to correctly define the task (e.g. initiative dashboards).

2. Make it easier to keep definition of task up-to-date (e.g. in-place editing),
3. Make it easier to see progress toward goal (bring code changes into the issue ala Github, metrics, completion checklists ala Trello (could integrate the “gates” idea into this).

Categories for collecting/tagging screenshots

- Dashboard
- Search
- Issue
- Comment
- Metadata editing
- Tool name

Tools

Trello
Description: Highly flexible, simple, customizable, fast, realtime updates, easy access to information
Problems trying to solve: easily communicate who’s working on what, what the highest priority projects are, what is the status of various projects. Not just for software, they intend it to be used for any type of project.
Patterns/metaphors: whiteboard covered with post-it notes, edit-in-place for everything, ajaxy, activity stream, drag-n-drop, quick drill-down between levels of abstraction

Unfuddle
Description: Fairly standard issue tracking
Problems trying to solve: Help software teams organize work against milestones.
Patterns/metaphors: Activity stream, priorities, milestones

Pivotal Tracker
Description: Agile PM tool. Drag n’ drop issues into priority.
Problems trying to solve: Simple agile management for software teams, estimate how long it’ll take for features to be built based on velocity estimate
Patterns/metaphors: agile metaphors - current/backlog/icebox queues for stories, story points, drag-n-drop, ajaxy, quick drill-down between levels of abstraction

Github
Description: Unique blend of code/branch management and issue tracking. Pull requests tool

is completely unique -- blends issue tracking with diffs of changed code and with code review
Problems trying to solve: Bring together in one place all the information needed by a contributor to help a maintainer integrate a pull request.
Patterns/Metaphors: Lots of faces, mashup of pull requests/code review/issue discussion, pretty standard otherwise.

Fogbugz
Description: Software issue tracker. Seems focused on formal traditional software development. Somewhat archaic clunky design.
Problems trying to solve: Keeping traditional software teams on track and provide good quality estimates for project completion
Patterns/Metaphors: evidence-based scheduling i.e. look at the data. Which seems like their only big insight/differentiator http://www.fogcreek.com/fogbugz/features/ Otherwise, seems pretty standard stuff.

Jira
Description: provides issue tracking and project tracking for software development teams
Problems trying to solve: Combines a clean, fast interface for capturing and organising issues with customisable workflows.
Patterns/Metaphors: ajaxy, lots of overlay, Individual forms for all actions (ex. Related form, Close Forms, all forms have a log/comment box for detail)