If You Build It, Will They Come?

A Collaborative Framework To Share IT Knowledge

Barbara Johnson
Jonathan Reed

Ask Yourself...

- How do you manage the constant stream of always-changing technical information needed to do your job?
- How can you minimize the number of new information repositories springing up within your organization?
- "Dive In! The water's great!"
  What will motivate your audience to participate in information contribution and sharing?

What Questions Did We Ask?

- What do you expect from a knowledge base?
- What would encourage you to use it?
  - What would encourage others to use it?
  - What would discourage you and/or others from using it?
- What's your "start page" for IT issues?
  Do you have one? Do you have your own?
- What's better, browsing or searching?
  - Browsing allows users to learn, explore topics
  - Searching allows people to get what they need now

In The Beginning...

- Static HTML pages
  - URLs changed with every re-org or site redesign
- FileMaker database with web server for Mac/PC stock answers
- Home-grown text-only system for UNIX environment
- High barriers to entry for all of these
  - Many groups chose to maintain their own repository (public and/or private)

Knowledge Centered Support

- Develop knowledge as part of problem-solving process
  - When you solve a user's problem, document it
- Knowledge is shareable and evolving
  - Exceptions in Support Industry (personal/tier2 contact info, software license codes, etc)
- Knowledge Base is key component

Is There A De Facto Standard?

- "No"
- What's out there?
  - home-grown solution
  - Wiki
  - CMS
  - Knowledge bases closely tied with ticket tracking system

Take One: Content Management System

- Commercial product (Solomon Retrieve)

- Flexibility with file types and access
- Not very end-user friendly for contributions; designed to be edited by a small group of people
- Difficulty integrating with local authentication and directory systems

Take Two: MediaWiki

- Free, open source
  - Lots of extensions, no need to re-invent the wheel
- Anyone who has used Wikipedia is familiar with it
- No granular access control
  - “Namespaces” were evolving and not quite what we wanted
- Searching is limited by search functionality of backend database
  - Wiki search behavior (page name vs full text) was confusing
- Evolving product, constant updates, API/functionality changes

Take Three: Confluence

- Commercial product (Atlassian)
- Good access control
  - Granularity (“Spaces”)
- Better searching
- Extensible API (Java and XML-RPC)
  - Large user base; contributed extensions
  - API still evolving
- Already in use elsewhere at MIT

Contributing And Using

- Who?
  - IT Staff
  - Community members
  - General public
- Why?
  - Because it's their job
    - To make their job easier
  - Because they want to share information
  - Because they need information

Barriers To Entry

- Wiki markup (or browser support for WYSIWYG editors)
- Time pressure (for frontline support, primarily)
- Different self-imposed “standards” for publication
- Unease with public information, even if there’s nothing inherently confidential
- “Ownership”: if I write about it, I might have to support it

Mitigation

- Deem contribution to be part of job description
- Get buy-in from organization that this will be **the** Knowledge Base
- Remove artificial or unnecessary barriers (wiki markup, browser issues)
- Encouragement, incentives

Lessons Learned

- It's harder to get contributors than we thought. It takes time.
- Did not set up a tracking mechanism up front. Can't tell who's looking at what. Some Hermes stats but not specific enough. Would like to implement Google Analytics.
- Get buy-in from decision makers to make "executive" decisions setting expectations for internal IS&T groups to contribute information.
- Important to minimize number of spaces, or separate collections of articles, within the knowledge base. Want to avoid information silos.

Lessons Learned (cont.)

- Need to be clear about what information goes where, e.g. website versus knowledge base. In the beginning we were seeing some duplication of information. Improving search functionality, to search both sources simultaneously, has helped to resolve this problem.
- Ongoing maintenance is required to keep content fresh.
- There's a need for an advocacy role to continue to inform community members about the knowledge base and it's usefulness. Demonstrate how easy it is to reuse solutions and help others in the community.

Success Stories

- The right people in the room. Project team had good cross section of real users. Sought input from support providers across campus.
- Project had definite end date and finite resources. Special one-time allocation of funds. Ongoing maintenance cost absorbed by Help Desk.
- Can now publish publicly viewable answers instantly.
- Hermes supports multiple data types, not just text.

Success Stories (cont.)

- Fosters collaboration and sharing of writing efforts. Example: Previously project and release teams had one assigned documentation person. Now everyone can contribute. Can use the documentation person's expertise more effectively as editor.
- Can instantly grant access to anyone who needs it, such as our student employees.
- Provides a direct feedback loop to article writers via comments functionality.
- Branding of articles, via logos and color schemes, visually indicates which subject matter areas are supported by IS&T.

Future Plans

- We need a link checker, to ensure accurate links within articles. We're actively researching tools that might work with Confluence.
- Implement Google Analytics to better understand usage patterns. Can help guide work in other areas of the support organization.
- Figure out how to mark "expert" topics. A visual identifier for articles with deep technical information. Perhaps an indicator similar to ski slope signage, e.g. black diamond difficulty.
- Work with established user groups and support groups. Schedule periodic refresher courses for searching and article writing.

Advice

- Capture what you're figuring out when you're figuring it out. (aka KCS)
- Let people solve their own problems.
- Don't try to know it all. Instead try to know where and how to find the answer.
- Reduce the number of places to look for information.
- Encourage less-than-perfect contributions.
- Minimize steps for access to knowledge base and authoring permission.
- Use familiar tools where possible, e.g. Confluence and Certificates. Both were already in use at MIT. Helped reduce cost and learning curve.

For More Information

- See Hermes article for presentation proposal and contact information, http://kb.mit.edu/confluence/x/Cgla
- Hermes Knowledge Base http://kb.mit.edu
- Barbara Johnson, bdoyle@mit.edu
- Jonathan Reed, jdreed@mit.edu