Practical Lessons for Applying Technology to Knowledge Dissemination

Susanna Cantor and William Savage
RTI International
Durham, NC USA

Gary Franceschini
La Trobe University
Melbourne, Australia

1 Abstract

One of the hardest activities in any large project is keeping all team members on the same page when it comes to expectations, processes and consistency of information.

When a project team includes both internal members and external partners, providing consistent information to all team members securely becomes difficult. Attachments to emails clog inboxes. Multiple document versions flying through cyberspace in crisscrossing email threads undermine effective document management and change control, injecting confusion into project operations.

Knowledge management begins with a central, easily accessible but secure means to share information in a timely manner and maintain it over the life of the project. In this paper, we will look at how the Knowledgebase Management Team (KB Team) for a large project at RTI International chose to develop a knowledgebase (KB) using the Drupal framework to distribute clear, concise and valid project information across geographic and organizational boundaries. [1] Using the Drupal site the KB Team developed as our example, we will explore the concept of a central KB, our implementation approach, our three main challenges to success, and the lessons we learned**.** The biggest lesson? Communicating knowledge can be the hardest part of managing knowledge.

Keywords

Knowledge Management, Communications, KGCM concepts, theories, models and methodologies, technologies, supporting systems, tools and techniques

2 Project Background

The RTI project was centered on a federally-funded, multi-component system using an Internet-accessible secure website with more than 40 staff serving approximately 700 system users. An essential component of this project was a help desk based at a sub-contractor's office in another state. The help desk served the website users – federal grantees who entered data about their programs. The help desk agents needed knowledge about the project to help the grantees use the website for performing data entry, data upload and download; running reports and obtaining current information about the project, such as a quarterly newsletter and other relevant information.

As sub-contractors, the help desk agents did not have access to RTI's internal network, where most of the project knowledge was electronically stored. Early in the project, we found that help desk agents needed more than the information provided in an FAQ and other documents posted to the website. They needed to be able to quickly access controlled, valid information and trust that it was current and correct.

3 Approach

The KB Team determined that the best way to provide trustworthy information to the help desk was to create an online centralized knowledge repository where operations and client support information could be made available for easy access by the entire project team.

The KB would serve two purposes:

- Centralize knowledge about the project system in a searchable repository.
- Provide a means to disseminate project information to all team members, regardless of work location and network access.

The KB would enable the help desk to offer better support to the grantees, and allow the technical group to share system operation and maintenance information, and the full project team to communicate consistent information.

The KB Team investigated several software options, and chose Drupal as the framework for the repository for its cost (free), its significant international user-base, its perceived ease of configuration and its ability to provide secure, role-specific content from one location. In addition, the team discovered an extensive user and developer community that actively supports each other and is available for consult if needed. All these attributes were attractive to a federally-funded project with limited resources to develop internal infrastructure.

Before we began configuring Drupal, we asked the entire project team, including members based with sub-contractors, to define requirements for the site.

At a high level, everyone agreed we needed a way to create, post, and manage content, and search posted content on role-based information pages. The pages would be deployed in stages as we had topics to post. We assumed we would reuse existing project information and that the various task teams would provide new content on a regular basis.

We then, as a group, defined system requirements for the framework, users, and the interface. Our help desk sub-contractor developed use cases to define how they would use the site.

Once the project team approved the requirements, the KB Team set out to configure the site.

We first downloaded and installed the Drupal core system, which resulted in a functional but empty website that supported site administration, content management tools, user management, security, and administrative reporting.

Next, we selected a theme, defining the basic layout and look, including display colors, navigation, menus, and buttons.

The Drupal community provides many modules, which are small, discreet, often powerful applications that can be added to a Drupal website. The KB Team worked hard to identify the modules to provide the desired functionality. Our Drupal configuration utilized approximately 35 modules, including Comment (allowing users to discuss content), Taxonomy (allowing users to quickly find content), Upload (allowing users to add content), and Notifications (allowing users to subscribe to content most critical to them).

Once the basic system, theme, and modules were established, the team performed various administrative actions. These included defining roles and taxonomy terms, and creating users with roles.

With a basic site configured, we:

- compiled a matrix of existing information and its location on the network share drive, which would comprise the initial KB
- identified the relevant taxonomy that would drive search results
- identified the roles that would need the information
- assigned one person to input the content.

Each piece of information was converted into a Drupal topic and its node or topic number noted in the matrix. This allowed us to track our progress and ensure we input the correct content for the release phase.

After we had input and verified the content, we released the KB to production.

Challenge 1: It’s not as easy as it looks.

As with any project that shows promise, standing up the Drupal site seemed to make sense. Drupal is highly configurable, with hundreds of modules that are easy to install and appeared simple to configure. Technical staff on the project planned to “figure it out” and, in a few weeks, create the vessel into which knowledge would flow.

The reality is that Drupal’s 49 module categories [2], each with multiple features and possible functionality, provide so many possibilities that it required time to research each one to determine which would provide the best features to build our KB.

The team’s hopes of quick configuration quickly faded into the reality that there was more to Drupal than met the eye, especially since we had no prior experience with it and no internal resources we could turn to, and we had to devote more time than anticipated to eventually achieve the results that we needed.

Because so many configurations were possible, there was no clear step-by-step configuration map that resulted in a finished site that met our needs in the timeframe we had allowed. We were also unsure if the modules had to be installed in a specific order (they do) or if there were dependencies that needed to be respected (there are).

Over the course of multiple sessions each week for several weeks we managed to build the basic site that could then be populated with content.

A small number of PDFs were uploaded as attachments to pages, but the bulk of the content was hard-coded in HTML using Drupal’s Create Content function. This required the skills of a documentation specialist with HTML coding abilities.

Once the content was input, it was verified by project team members and the system was tested before we could declare it ready to use.

Challenge 2: Why use a new system when ours works just fine?

Although the entire project team, which included both RTI and external partners, agreed the KB was a good idea, it became clear over time that old habits die hard. RTI had a process for producing and circulating project information, and our sub-contractor had its own process. Each found it hard to shift to a new way of sharing information, and using the KB rather than internally stored documents proved a challenge for the entire team.

Project team members also had to learn the new system. The help desk could not shut down for training; it had to continue managing calls from website users. In the heat of battle, the sub-contractor staff fell back on their known and familiar processes, which included maintaining a local repository of information to answer the questions.

On the RTI side, staff focused on meeting project deadlines and did not have the bandwidth to learn a new process or system, despite the fact that the KB Team provided training and one-on-one support.

Plans to control project documentation by managing postings to the Drupal site fell by the wayside as the project team fell back into individual habits to get the work out the door.

Even the staff tasked with creating the KB found that other project duties diverted their attention to continuous monitoring of the site and its contents.

Challenge 3: What, you want me to maintain all that stuff?

Initially, the KB implementation team gathered project operational documentation and set out to post it to the Drupal site. The task was time-consuming but routine: turn current project information into topics on the site, reference those topics to role-based pages and publish them so the help desk and other project task teams could use it.

Once the initial content was loaded, we created a process to add new content and appointed content administrators who, on their own, would initiate posting of important information. We also set up a workflow for editing and linking the content to the correct page. However, the task eventually fell to one or two people to actually post content, and the idea of a content team distributed across the full project team slowly devolved.

A related challenge was reviewing and updating the posted content. This task proved difficult because the project had not initially allocated resources for a dedicated webmaster to regularly review the site for outdated items.

Eventually, we assigned a webmaster who managed the technical and content areas of the site. He also created a Drupal “cookbook” that provides step-by-step instructions on how we created the site, including the sequence of the configuration and the settings we used. The document was to be used by other RTI projects who were interested in standing up a Drupal site. However, when our webmaster left RTI, he took with him the detailed knowledge required for our continued Drupal success.

4 Lessons Learned

Our experience taught us valuable lessons in managing knowledge and communicating it to others who need it.

More Choices

The first lesson is this: examine as many KB framework choices as you can. There are many, and it is critical to understand as much as you can about the complexities and trade-offs of the tools being considered. Drupal is only one of several candidate tools.

Since the KB Team built the Drupal site, other RTI projects have employed different tools to serve the same purpose. LifeRay, SharePoint and MadCap Flare have been used as alternatives for KB frameworks.

LifeRay: LifeRay, an enterprise web platform, [3] has enjoyed limited use on some RTI projects. In one example project, LifeRay was used to provide an out-of-the-box framework to support entering and sharing grant progress information, with a supporting workflow process to manage review and acceptance of submitted information. The framework was enhanced with custom Java controls, and the resulting system was deployed onto an Oracle database.

SharePoint: RTI has set up a central SharePoint portal and can configure project-specific sites as needed. SharePoint provides document management and control, is scalable and can be fully integrated with business intelligence tools. [4] It supports .NET, jQuery and Silverlight and has available a vast array of resources in the form of consultants, on-line information, books and plug-ins. [4]

Projects, especially those that work with sub-contractors who don't have access to RTI's internal networks, use SharePoint to pass documents around the team for review, compile monthly contract compliance reports, post project team contact information and keep a calendar of important project dates. Some also devise ticketing system workflows that track project activities through SharePoint.

At RTI, about 80 different projects [4] maintain information and collaboration sites and sub-sites, some of which are accessed by non-RTI project team members. RTI itself uses SharePoint for KBs, document storage, calendars and discussion boards.

Madcap Flare: Documentation specialists in RTI's Research Computing Division have used Flare [5] for a number of years to create user information that can be posted online. In Flare 8, the current version, new features support populating a KB using content created and managed through this software.

Basics for a KB include using Flare to produce the content and output it in a searchable, indexed packet; using companion software Madcap Contributor to allow subject matter experts to review content and in some instances create it; and maintaining all the files in a source control repository, such as Subversion. [6]

Flare also allows content to be imported from other sources, including Microsoft Word, which our Drupal site was not configured to do. This makes it easier to start populating a KB when the content can flow in and doesn't have to be hard-coded. And, certain configuration files, such as CSS, skins and templates, can be imported by and shared with other Flare projects.

These are just three possible alternatives to Drupal that have proven useful in centralizing information for communicating knowledge. If anything, this lesson taught us to ask better questions about usability and ease of workflow once the KB is built. Because these alternatives were easier for RTI projects to use, some have moved away from considering Drupal as a knowledge management framework.

Content Life Cycle

Plan to regularly review and update your KB content efficiently and on a consistent schedule. If you set up a KB, you should commit to maintaining the content, and putting it on a life cycle with regular review and updates. Regular review will help remove invalid content and add new topics as needed. It will keep things fresh.

Content in our project KB was valid at the moment it was posted, but as the project progressed, information evolved at a rapid pace. Even with a dedicated webmaster, we found we lagged behind in the updates.

For the next KB, as with any project document, we plan to review all content each year by dividing it into quarterly update increments. We will assign each topic a review period, and during that period, designated authors or content managers will review or update any information that has changed. We also will set up Microsoft Outlook reminders so we do not forget.

The team should also control versioning of all topics so that the project is using the correct, most recent information. With each annual review, the team will advance the version number a full-number increment and track the expiration dates so we can easily do it all over again the next year.

Of course, urgent updates can be made at any time. But if you have a life cycle mapped out and implemented, you can ensure that all your content can remain valid – and useable – over time.

Top-Down Compliance

Finally, we learned that encouraging the project team to use the KB starts at the top and flows down to the rest of the team.

Engage project leadership to enforce the "use the KB" rule. Even though project leadership endorsed the concept of a KB and we developed a standard operating procedure governing its maintenance and use, it fell to one person to enforce its use. And with project leadership focused on meeting contractual obligations, we lacked a clear approach to reinforcing the need for and use of the centralized repository.

Engaged project leadership can help guide team members who have difficulty remembering the message and benefits.

5 Acknowledgements

The authors appreciate the assistance of:

- Dan Reid, RTI, for providing details of RTI's use of SharePoint.
- Craig Hollingsworth, RTI, for editing assistance.

6 References

[1] http://drupal.org/

[2] http://drupal.org/download

[3] http://www.liferay.com/

[4] D. Reid, K. Britt, S. Kalinga, SharePoint Overview (selected slides), RTI International, 2012.

[5] http://www.madcapsoftware.com/

[6] N. Perlin, Using Flare as a Content Management System, MadCap Flare Webinar recorded April 17, 2012.