Introduction

This white paper results from a charge issued by the Library Executive Group to provide an assessment and roadmap for Cornell University Library (CUL) to create an effective and efficient web development and management strategy. This white paper summarizes the current state of web architecture and development practices at CUL, and makes recommendations for achieving sustainable efficiencies in the existing web environment, as well as with new web based projects going forward. Input for this paper was gathered from a broad range of stakeholders across the Library.

Web architecture is defined as the approach to the design, specification and planning of websites. The approach includes the technical, aesthetic and functional aspects of Library websites as a means to meet the needs of our users. CUL's current web architecture and development model is based upon dozens of sites and hundreds of pages that incorporate a variety of technologies and design aesthetics.

While it is difficult to quantify the value of web projects, it's a given that staff time is both the highest organizational cost and the rarest organizational resource. Saving this time and investing it more wisely when utilized should be one of an organization's highest priorities, and should be constantly re-evaluated. This paper will seek to identify current and future strategies for reducing web architecture related costs at CUL, while also focusing on a realistic model of sustainability.

Current Efforts

IT and related activities are a large cost center in the organization and typically, the largest portion of those IT costs is staff time spent writing, debugging and eventually, maintaining custom software. Areas that require the most staff time are easy to identify and include:

- integrating two or more systems,
- maintaining legacy systems in production, including addressing security vulnerabilities,
- task switching between many active and legacy projects,
- building new custom projects, and
- customizing what already exists.

Therefore, the most effective way to reduce costs on IT projects is to reduce the amount of custom software written. In an environment where a cost/benefit analysis can be done, a high cost/low value project will simply not be done. For green-lit projects, IT costs (including secondary costs such as addressing security vulnerabilities) can be reduced through strategies that emphasize standardized infrastructure over custom code. Such strategies include:
● Using commercial off-the-shelf (COTS) software and cloud services,
● Limiting customization of COTS software and cloud services,
● Using frameworks and third party components,
● When custom software must be created, maximizing its re-use.

The following are some examples of existing strategies.

*Policies that encourage use of the most appropriate content management system.*

The Library has invested in several generalized content management systems (CMS) which should be evaluated against client requirements to choose the option that best leverages existing infrastructure so technical tasks focus as much as possible on the unique requirements of the project.

<table>
<thead>
<tr>
<th>CMS Platform</th>
<th>Indicators</th>
<th>IT Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal</td>
<td>● Site is currently in an unsupported/end of lifecycle platform.</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>● External facing site.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Requires highly customized graphic design/interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Highly customized dynamic functionality.</td>
<td></td>
</tr>
<tr>
<td>Confluence</td>
<td>● Internally facing site.</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>● Limited design/customization needs</td>
<td></td>
</tr>
<tr>
<td>Dreamweaver/Static</td>
<td>● No significant dynamic functionality required.</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>● Very small site.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Infrequently updated.</td>
<td></td>
</tr>
<tr>
<td>Libguides</td>
<td>● Tutorials (including course related materials) about library services,</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>authored by librarians.</td>
<td></td>
</tr>
<tr>
<td>Blackboard/Moodle</td>
<td>● Other course related material</td>
<td>Low</td>
</tr>
<tr>
<td>WordPress</td>
<td>● Blogging as primary activity.</td>
<td>Low</td>
</tr>
</tbody>
</table>

*Investment in a limited number of frameworks.*

There is a tension between taking advantage of new technologies to create more efficient and innovative services, and limiting the number of technologies in use to hold training and
maintenance costs in check. The Library has done well choosing frameworks to invest in that are used generally over many services.

In recent years, Drupal has become our primary content management and Web publishing platform, and more recently Ruby on Rails (RoR) has been adopted for collection management and searching. We have also adopted jQuery for creating dynamic AJAX interfaces and Bootstrap for attractive, responsive interfaces. These tools each have many third party components built for them to solve common needs, allowing library staff to focus more time building only what is truly unique to the Library, and to do so more efficiently. Widespread use of these frameworks on many projects enables more flexible support from staff cross-training.

Migration from physical servers to virtual machines.

Moving to virtual machines has resulted in more standardized environments, and more efficient provisioning of server resources. It also has the effect that the Library is responsible for less of the architecture stack than it has been in the past, significantly lowering maintenance costs.

Using tools to standardize deployments.

Configuring third party deployment tools (such as Puppet, Drush and Capistrano) allows us to come up to speed more quickly on new projects, and creates standardized environments that are easier to maintain. As the Library has settled on these standard tools, we have also effectively increased the capacity for emergency response to critical system issues by having redundant skillsets in system management across the organization. Contrast this streamlined, centralized approach to the more distant past when individual staff were responsible for individual physical servers that were all configured uniquely.

Collaborations outside the organization, most notably, 2CUL, the Mellon Linked Data for Libraries Grant collaboration with Harvard and Stanford, and the Hydra Project

Collaboration with other organizations within Cornell or other external institutions is a clear investment of time that minimizes the costs and risks associated with custom software while maximizing the benefits. We have already seen significant benefits from our collaboration with the Hydra Partnership (http://projecthydra.org) during the development of our Blacklight-based catalog search system.

Releasing software under an open source license is a way to encourage ad-hoc collaboration, and has been done with some success in the organization, for example, with Vivo, Project Euclid and Handle Web Services, and will likely continue with the work currently being done with Blacklight and Hydra. This approach may have more up-front costs, not only taking on more of the initial development ourselves, but also creating basic documentation, installers, etc., with no guarantee of getting the involvement of others with shared goals. However, this approach may be preferred with more experimental endeavors, and has the added benefits of leveraging relationships with peer institutions, which may lead to the ability to collaborate with partner
organizations and provide more influence on vendors to deliver better solutions for our shared benefit. Additionally, the reputational benefit of giving back to the library community through code sharing should not be overlooked. While this approach has been successful thus far, it will be important to monitor the costs and benefits of this method as time goes on to ensure continued benefit to the organization.

Better focus through an IT Project Inventory, LibExec prioritization and sprints.

In 2012 an authoritative IT Project Inventory used by LibExec to determine organizational priorities was created. This along with the adoption of agile development methodologies that include the practice of development iterations or “sprints”, has allowed us to reduce cross-project conflicts, costly task switching, and put more focused effort on fewer projects, allowing us to complete our most important projects much more quickly.

Common features and services across sites.

Examples include the Curated List of Library Resources (CuLLR), a locally developed project that was generalized across several “virtual libraries,” and various library web services, which provide convenient access to standardized data such as reserves and Library hours that are leveraged flexibly across a number of library sites.

Partnering with Usability Working Group

Due in part to the “cross pollination” of the Usability Working Group with user interface designers and web developers, awareness of usability best practices has greatly improved amongst the staff working on web projects in the Library. Additionally staff have been able to easily get web projects on the calendar of the Usability Working Group, and the Working Group has routinely included designers and developers as stakeholders. This also allows for testing of new features in early stages of development, at a time when it’s less costly to make changes.

Recommendations

Require all web projects to be submitted to the IT Project Inventory for prioritization.

Prioritizing and project planning are paramount to value realization. Too many competing projects result in non-productive fragmentation of staff time on projects and the added overhead of task-switching. Although it may be counter-intuitive, more focused attention on fewer projects means that although a given project may begin later, it is more likely to finish sooner. Projects and tasks must also be judged relative to other projects. Software by its nature is always incomplete, and every project reaches a point of diminishing returns, where time may be better spent on another project with tasks of higher relative value. Completing projects that meet organization-wide priorities brings more return on investment than starting multiple lower priority projects.
Long range project planning and prioritization versus working through projects on an ad hoc basis, affords the opportunity to better utilize staff skills and time, as well as the chance for planned repurposing of code, technology stacks and design themes allowing teams to operate in a more efficient manner. Reusing existing components allows for shorter development cycles, reduced ramp up periods and long term savings in the form of fewer customized maintenance needs.

Enhance the effectiveness of LibExec prioritization by ensuring that decision making responsibility is at the appropriate level.

The IT project list that LibExec prioritizes addresses the question of “what” the organization should be doing, but “how” it should be done should be left to those charged with implementing the projects. Making decisions at higher levels of the organization than are necessary increases organizational costs, such as administrative overhead and lost opportunity costs, while lowering the lifetime value of any positive outcome of such decisions.

The impact of new projects on existing web architecture must be considered before being approved.

While new technologies must be introduced over time to take advantage of advances that can lead to innovative new services, this is done at a cost in staff training and new maintenance needs. This investment should only be made for new technologies that are likely to be generalized, such as more modern frameworks, as opposed to “one-off” needs. Common sources of arbitrarily chosen technologies include student projects and other outsourced IT effort.

Project implementers must make infrastructure, standards, and policies an explicit part of the planning, development and maintenance.

A high-quality product must be based on the best industry standards for addressing infrastructure, or “cross-cutting” concerns, such as:

- security and privacy,
- performance,
- accessibility,
- device responsiveness,
- branding and consistent style guidelines, and
- usability.

Traditionally, these aspects have been introduced into projects by system administrators, application developers and user experience designers. There is a direct and substantial time commitment to plan, implement and maintain these aspects of a project that must be considered in project timetables and budgets. And once again, it should be noted that the use of existing products and frameworks can substantially reduce these infrastructure costs.
Maintenance costs must be estimated and considered, and a sponsor identified, before a project is approved.

Over the lifetime of a system, maintenance costs typically dwarf development costs and they generally increase over the life of the project. Estimating these costs is particularly important for projects with set funding such as grants and other externally supported projects. While technological innovation is valued, experimental projects put into a production setting have real long-term costs associated with them, which should be thoroughly considered, and if possible, provisioned for at the outset. It’s important to identify a sponsor to provide leadership and delegate ongoing resources for the project.

Perform regular strategic reviews—at least annually—not only of new projects and their relative priorities that might be started, but legacy systems and their relative value, which might be ended.

Staff time is a scarce resource compared to the number of IT projects. Once in production, a development project becomes a legacy service, with maintenance costs. If projects are put into production more rapidly than they are taken out of production, staff capacity leaks until it is completely drained. When staff time is at maximum capacity, new work can only be added if offset by increasing capacity or removing existing work of comparable scope. Additionally, there is value in considering sunset dates for projects that may serve a short term need, and/or acknowledging the possibility of dark archiving rarely accessed digital collections.

Potential questions to consider during an annual review include, but are not limited to:
- Does this project still meet the need it was intended to?
- Is this still the best way to meet that need?
- Is there a better way to meet that need, and what is the cost of switching, or not switching?
- What are the consequences of discontinuing the project?
- Does this project meet current standards and policy objectives? (i.e. accessibility, mobile first/responsive design, etc…)
- What work can be effectively outsourced?

Carefully consider Applied Programming Interfaces (APIs) when evaluating third party software.

The costs associated with using COTS and cloud services increase when we are expected to be able to interact with these systems for custom functionality, possibly interacting with multiple such systems simultaneously. We have been doing a better job in recent years of including this in our evaluations of third party software, but particularly if a technical person is not directly involved in such an evaluation, the following questions need to be asked and considered:

- Do API’s for the product exist?
- Do they supply the desired functionality we anticipate needing?
- Are they documented?
- What standards do they support (RSS, Z39.50, OAI-PMH, REST, JSON, etc.)?
- Are they robust, secure, and efficient?
- Do they have usage quotas? (Cloud services.)

Typically, API's are Web Services, but in their absence, direct database access or client software may be acceptable. If API's do not exist, or they are not robust enough to serve our needs, then this automatically must be considered a high risk to any project that may need the data contained or managed in the third party software. Including staff with development and user experience backgrounds in projects well before they include a broad technical component will help get these questions answered early in the process.

*Make strategically important systems service oriented.*

Where API's are lacking in our own systems or third party software, we may seek to build our own web service solutions to integrate these systems into our environment. Current examples already cited include CuLLR, library hours and course reserves.

An API is a contract between two or more systems and must remain stable to be useful. As an organization, we need to recognize these services as mission-critical parts of our legacy infrastructure and support them accordingly.

*Encourage consistent theming and branding across Library sites.*

Custom theming of sites is a high cost activity that often results in sub-optimized gains and can have other negative consequences for users trying to navigate library sites. Examples of this include the custom designs of digital collection sites and other library websites where design is undertaken without taking into account reliable user needs data. Library sites should be created based on user needs research, simultaneously bearing in mind the unique needs of distinct user communities, while also being mindful of the benefits of maintaining design continuity across Library websites. Common theming sends a unified message to users of Library websites, and connects the broad distribution of Library sites and services to provide a cohesive brand message and ease of use. Common theming should be accomplished by establishing a graphic standard, which guides the user interface design of the Library’s sites. This does not translate to all sites looking identical, but rather maintaining a consistent feel to present a cohesive image of the Library, and consistent navigation to improve user experience.

A review of peer institutions (Stanford, Columbia, Yale, and Dartmouth) demonstrates the benefits of a single library web portal and a unified look and feel for all library websites. Presenting the library as a unified entity puts forth a strong message which encourages patrons to respond positively to unique CUL branding. Additionally, the unified look and feel offers users the ease of learning a single mapping of features and functions regardless of the library website they are interacting with. We know from user studies that students and
faculty routinely interact with more than just one library website (e.g. that of the library supporting the department or college they are directly affiliated with).

For existing sites, a redesign merely to enforce consistency is obviously high cost, and not recommended solely for the sake of consistency, but when a redesign is planned for other reasons, consistency, coupled with the need to respond to user needs, should be a priority.

*Involve CUL UX Team in all Web Based Projects in the Libraries*

Websites and web applications have evolved from a one-way static medium into a rich and interactive experience for users. Bearing in mind that a website or application is only as successful as the positive experience of our users, CUL’s UX team focuses on how users feel when interacting with our systems, and they have the background to allow them to accurately evaluate the overall user satisfaction, from ease of use to task efficiency and perceived value of a system. The goal is to build on the current success of having UX staff involved on a consultant basis with projects such as the upgrade to LibGuides 2.0, the Hydra pilot, and exploration of Omeka.

One member of the UX team is designated to interact with web project working groups/implementation teams on a consultant basis to provide support in the following areas:

- advocate for users - both public facing and administrative components of websites; are systems easy to use and do they play well with existing systems?
- establish a more unified user experience among CUL web sites, e.g., consistency in branding, navigation, labels, and search experience;
- ensure consistency in code and follow web standards; develop code that is easy for any other UX member to work with as the need arises;
- coordinate with CUL usability working group to arrange user studies of the system;
- certify compliance with Cornell accessibility guidelines and standards;
- act as a liaison between the working group and other CUL entities, such as: developers/programmers, usability working group, and library communications;
- help create a long term plan for maintenance and ongoing support of software/tools;
- assist with outreach and product demos - sharing the tools and concepts behind creating usable websites.

*Usability and usage data should inform development of websites and digital collections.*

A proposed project should have as part of its proposal clear metrics to determine the quantitative or qualitative of the project. Once in production, the project sponsors should evaluate these metrics regularly and report back to LEG on the effectiveness of the project to determine if the effort is successful and should be continued.

Usage data on Library websites is collected as a matter of course from web analytics. This unbiased user behavior data can be used, along with results of user testing, to inform design
and development improvements and to direct maintenance efforts on the Library’s most frequently accessed web assets. This data can also provide tracking data for user behavior, such as where users are reaching our resources from. Click analytics data can also show how users navigate web pages, and help identify potential problems with navigation and functionality.

Be committed to iterative design/development model.

The iterative approach consists of implementing parts of a system at regular intervals. Working in iterations offers many benefits, including allowing the Library to be more nimble in responding to the needs of users; making it easier to detect risks associated with a given project earlier in the process; allowing usability testing to be performed on individual iterations before they are implemented; and enabling resources to be committed for more concentrated and more predictable periods of time.

Continue using cross functional project teams

The development and deployment of the Library’s Blacklight catalog has been well received by Library staff and patrons, an experience not mirrored by many of our peer institutions. Success of CUL’s project is due in large part to the diverse, cross functional team instrumental in the forward progress of this new, highly complex system.

This approach relies heavily on three distinct teams working in close concert to gather requirements, design, develop and test the product. The user representative team is comprised of representatives from across the CUL community, with particular emphasis on including staff responsible for reference and instruction, as well as staff from unit libraries whose patrons have unique requirements of library systems and services. This team is responsible for reviewing the various iterations of the product, testing the product and providing timely feedback. These individuals represent the primary stakeholders for the project and, as such, provide many of the user stories which represent project requirements. The application developer team is responsible for interpreting, estimating and implementing user requirements in software. The user experience/interface team is responsible for providing the front end design in concert with the infrastructure provided by the developers and based on input from the user representatives, results of usability studies.

By combining forces of these three groups this team has been able to produce a product which represents the broad range of needs within the greater Library community. This success can be replicated on future IT/web projects by creating similar teams, scaled to the size of the project and composed of technical, user experience, and public services staff. Other Library stakeholders (i.e. catalogers, electronic resource management, and communications & assessment staff) should be pulled into the project for limited engagements as their expertise is needed by the team.
**Conclusion**

This paper has established that there are many efforts at CUL already contributing to effective management of our web architecture and the associated costs. The recommendations contained herein form the next logical steps in the evolution of CUL’s web presence. The suggestions put forth affirm the importance of supporting the Library’s web architecture, and, if adopted, they will increase web development capacity and afford better long-term sustainability for the Library’s web projects. The paper also establishes working models to improve accountability, collaboration and efficiency, and to increase the emphasis on user and data driven design and development, allowing the Library to better respond to the needs of users.
Appendix

Reviewers/Contributors

Nick Cappadona
Gabriela Castro Gessner
Adam Chandler
Kathy Chiang
Jennifer Colt
Jesse Koennecke
Maureen Morris
Tracey Snyder
Gail Steinhart
Melissa Wallace