Editorial Board Thoughts: Content and Functionality: Know When to Buy 'Em, Know When to Code 'Em[1]

Kenneth J. Varnum[2]

We in library technology live in interesting times, though not those of these apocryphal curse. No, these are interesting times in the best possible way. Where once there was a paucity of choice in interfaces and content, we have arrived at a time when a range of competing and valid choices exists for just about any particular technology need. Data and functionality of actual utility to libraries are increasingly available not just through proprietary interfaces, but also through APIs (Application Programming Interfaces) that are ready to be consumed by locally developed applications. This has expanded the opportunity for libraries to respond more thoughtfully and strategically to local needs and circumstances than ever before. Libraries are faced with an actual, rather than hypothetical, choice between building or buying fundamental user interfaces and systems

As the internet has evolved, and coding has become more central to the skillset of many libraries, the capability of libraries to seriously consider building their own interfaces has grown. How does a technologically capable library make the decision to buy a complete system or build its own interface to existing data? The process can be decided using a range of criteria that can help define the library's need for a locally managed solution. We'll start by discussing technological capabilities needed to take on almost any development project, then define three criteria, and finally discuss the circumstances in which a build solution might be appropriate. The goal is outline a process for deciding when it make more sense to buy both the interface and the content, to build one or the other locally, or to build both.

Criterion 0: What are the short- and long-term technological capabilities of the library*?*

Clearly, the first point of consideration is whether the institution has the capacity to manage application development and user research. The short-term answer may be no, but the long-term answer -- one based on the library's strategic direction -- may be that these skills are needed to meet the library's goals or strategic vision. One project may not be enough to tip the scales, but if the library is continually deciding if the immediate project under discussion is the one to change the balance, then perhaps the answer is that it's time to invest in new skillsets and capabilities.

There are actually several skillsets needed to undertake development projects. Individuals with coding skills are needed to adapt existing open-source software to the library's needs — it is a rare

[1] With apologies to Kenny Rogers

[2]**Kenneth J. Varnum** (varnum@umich.edu), a member of the *ITAL* Editorial Board, is Senior Program Manager for Discovery, Delivery, and Library Analytics at the University of Michigan Library, Ann Arbor, MI.

open-source project that does *exactly* what a library needs it to do, with connectors to all the same data sources and library management tools already perfectly configured by somebody else — but that is not sufficient. A library also needs people with user interface and user research skills ensure that the application meets at least the critical needs of its own user community, and does so with language and cues that match user expectations.

Even if there is not a permanent capability on the library's staff, development can take place with contract services. If this is the option selected, a library would do well to make sure that staff are sufficiently trained to make minor updates to interfaces and applications, or that a longer-term arrangement is made for ongoing maintenance and updates.

Criterion 1: What is the need to customize interactions to local situations?

Most, but not all, applications offer opportunities to match interface features and functionality with local user needs. The more interactive and core to the library's service model the tool is, the more likely the tool is to benefit from customization. For example, a proxy server -- technology that allows an authenticated user to access licensed content as if she were in the physical library or within a campus on a defined network -- has little or no user interface. There is little need to customize the tool to meet user needs, beyond ensuring the list of online resources and URLs subject to being proxied is up to date. There really aren't any particularly useful APIs to consumer and reproduce elsewhere, and there are easier ways to build an A-Z list of licensed content than harvesting the proxy server's configuration lists.

In contrast, the link resolver -- technology that takes a citation formatted according to the OpenURL standard and returns a list of appropriate full-text destinations to which the library has licensed access -- may well be worth bringing in house. Some vendors offer their software to be run locally, while others provide API access to the metadata. At my institution, we used the APIs Serials Solutions makes available for its 360 Link API to build our own interface using the open-source Umlaut software. (See https://mgetit.lib.umich.edu/). Why go to the trouble of recreating an interface? For several reasons, some of which (understanding user behaviors and maintaining control over user data to the extent practical) I'll touch on in the following two sections. The main reason centered on providing a user interface consistent with the rest of our web presence, offering integrations to our document delivery service, and a way to contact our online chat service, and a way to report problem links directly to the library when the full text links provided by the system do no work. While these features are generally available through vendor interfaces, the user experience is hard to make consistent with other services we offer.

Criterion 2: What are the needs for integration with other systems from different providers?

Integrations can run in two directions: from the system under consideration to existing library or campus/community tools, and from those environmental tools to the library. When thinking about the buy-or-build decision, understanding the scope of these integrations up front is important. If all of the tools or services that need to consume information from or provide information to your

https://doi.org/10.6017/ital.v36i3.10087

system rely on well-defined standards that are broadly implemented, this criterion may be a wash; there may not be an inherent advantage to building or buying based on data exchange.

If, however, the other systems are themselves tricky to work with, relying on inputs or providing outputs in a non-standard or idiosyncratic way, this situation may swing the pendulum toward building the system yourself so you can manage. For example, many course management systems on academic campuses can consume and provide data using the LTI [Learning Tools Interoperability] standard for data exchange. Many traditional library applications do, as well, so if a library using an LTI-compliant system needs to provide course reserves reading lists to the course management system, this is a ready-made way to make that information available.

At the other extreme, bringing registrar's data into a library catalog -- to know who is in what courses to provide those patrons with an appropriate reference librarian contact for a particular subject, or access to a reading list through a course reserves system -- may only be possible through customized applications to read non-standard data. In this case, to provide the desired level of service to the campus, the library may need to build local applications.

Criterion 3: Who manages confidentiality or privacy of user interactions?

A final, and increasingly significant, criterion to consider is where the library believes responsibility for patron data and information seeking behavior to reside. Notwithstanding contractual or licensing obligations taken on by library vendors, the risk of inadvertent exposure or intentional sharing of user interactions is always present. One advantage of building local systems to interact with vendor systems (link resolvers, discovery platforms, etc.) is that vendor does not have access to the end-user's IP address or any other personally identifying information. The vendor only sees a request coming from the library's application; all requests are equal and undifferentiated. Of course, once users access the target item they are seeking (an online journal, database, etc.), that particular vendor's site has access to that information. For libraries concerned about user privacy, the risk of exposure is somewhat mitigated by managing the discovery or access layer in-house -- and deciding to maintain a level of user information that suits that particular library's comfort level -- and potentially minimizing the single point of failure for breaches.

At the same time, such a decision puts more responsibility on the library or its parent information technology organization to protect data from exposure. Some libraries feel they can handle this responsibility -- either by careful protection of the data, or by not collecting and storing it in the first place -- in a way that library vendors cannot.

Concluding Thoughts

Making the buy-or-build decision is not straightforward; the criteria described here are not the only ones a library might wish to consider, but they are common ones with the greatest ramifications. Putting the decision process into a framework can help a library make consistent

decisions over time, enabling it to focus on the projects and systems that are most important to the library and its community (a campus, a town, or company).

https://doi.org/10.6017/ital.v36i3.10087