cottage labs

www.cottagelabs.com

Software Selection Methodology (SSM) for Libraries

Decision Support Tool

Cottage Labs on behalf of LYRASIS, 28th October 2011
http://cottagelabs.com/

1. Intro

Choosing the right software for use in a library can be difficult when faced with the plethora of options and versions of library solutions available. Selecting the best software can seem impossible as variables often do not compare like for like and can be dependent on a multitude of tangential factors.

This document highlights the benefits derived from a successful Software Selection Methodology (SSM), and breaks down the various criteria and steps for choosing software. Although it is focused primarily on selecting software for use in libraries, the principles apply across all software selection scenarios. The methodical, systematic approach outlined provides the tools to compare Open Source and commercial solutions and aims to support the reader in choosing the right solution for their library.

Target Audience

This document is designed for those involved in the selection process, use and development of software solutions aimed at tackling library needs. This could range from Systems librarians to Technical Services department heads. This document is applicable to those without detailed technical knowledge but a level of knowledge about library systems is assumed.

This tool is intended as an aid to stimulate discussion and promote engagement among library policy makers or those involved in the process of acquiring or maintaining systems to support a libraries function. Since this scope is very wide this methodology does not aim to be prescriptive but rather to raise the relevant issues that should be considered when approaching a software selection process.

2. What is SSM?

2.1.1 Objectives

A Software Selection Methodology (SSM) provides a framework to analyze different software solutions and reach a decision on which to implement. Since software is integral to the operation of nearly all organizations, choosing the correction solutions is vital. A variety of processes have been developed to aid decision makers and these are generally structured around consideration of key variables or criteria.

When embarking on the process of selecting software it is important that all stakeholders have a clear idea of what they are aiming to achieve.

An SSM aims to:

- **Clarify the business goals and requirement for the new software:** By asking the right questions it should become clear exactly what is or is not needed by your library.
- **Create a suitable shortlist:** Having considered what is needed the SSM process should lead to a shortlist of options. This may be standalone software packages or combinations of features and a plan to integrate them together.
- **Analyze shortlist and select the best suited software:** From this list a solution (or combination of solutions) must be chosen to implement
- **Justify the selection to all users:** The process should be inclusive of all users so that they understand the process and are aware of the role the software plays in the successful operation of the library.

2.1.2 Critical Success Factors (CSFs)

In order to achieve the goal of a successful implementation it is important that there is a method to measure how the software will function in the library. Critical Success Factors help you identify the key metrics to follow and help prevent difficulties early on.

2.1.3 Benefits of Using a Selection Methodology

By following a well defined process and making sure that all stakeholders are aware of the decisions being made you can expect to realize several benefits.

A well thought out SSM will let you:

- Reach a decision quicker – save time (money) at the selection phase
- Avoid wasting money on unnecessary software and/or licenses
- Employ well purposed tools
- Be part of a relevant community of professionals
- Easily demonstrate value to management – a clear process is easier to explain and may expedite future acquisitions

- Be able to get targeted support if software is designed for your purpose

2.2 Existing Methodologies

There are a wide variety of SSMs available to aid the software selection process. Most of these advocate a weighting of priorities based on pre-defined criteria, however they vary in the way they go about this and the criteria they employ.

Before starting the process it is worth considering what other methodologies are relevant to your situation. If there have been previous software selection processes at your library, how were they organized?

Many SSMs are focused on particular software types or niche use cases and employ a more statistical approach to compare the variables. Where there is a well-defined marketplace of solutions there is normally a more streamlined comparison process as different packages tend to compete on key features and it is therefore easier to compare like for like.

OSS Methodologies

Open Source solutions often require a different approach when selecting solutions. It can be important to have a methodology that is well suited to the characteristics of this type of software.

It is, for instance, particularly important to precisely examine the constraints and risks specific to open source software. Since the open source field is very rich and has a very broad scope, it is important to employ a method that compares the community and innovation aspects of a given solution in tandem with the technical, functional and strategic requirements.

For more details see the appendix on existing methodologies [link]

2.3 Library Issues

Libraries face many unique issues when it comes to the software that they use. Due to the scale and scope of the information stored by libraries and the many different types of organizations that libraries serve there are a huge range of potential issues.

Some of the issues that libraries need to consider when selecting software include:

Software Licenses
The mission of most libraries is to disseminate knowledge and the software you use should align with the aims of your organization as a whole. You should take care that licenses do not restrict the abilities of users to carry out the goals of the library or contravene any strategy on freedom of access.

The rise of Open Source Software (OSS) has meant that libraries can now choose whether to use commercial (proprietary) solutions where the code is closed or open solutions that allow for greater customization. In general a company supports proprietary solutions, and a community of developers supports OSS.

As many libraries are publicly funded there may be a need to ensure that the code base is open and accessible by the general public. You should check what limitations will be placed on you by the various software options under consideration.

In deciding, which is right for your library, you will need to consider the costs, flexibility, functionality and support. For more details see the section on identifying requirements [link]

Data Licenses
Most libraries will deal with some content that is covered by copyright or other legislation that governs how it may be disseminated. The software you select must be capable of respecting these requirements and protecting sensitive personal data or metadata. It is important to consider how you can ensure that sensitive or legally protected data is stored securely.

Data Formats and Standards
Libraries rely heavily on well-defined standards for metadata creation and sharing. Selected software should be capable of respecting these standards and any data imported and exported through the system, as well as through its internal processes.

At the same time, the software should be sufficiently flexible to cope with changes in metadata usage as library needs develop and grow. Open source software may offer considerable benefits here, especially if supported by a community engaged in similar activities to the library.

Hosted / Software as a Service Models
Many commercial vendors in the library sphere are now offering hosted services, with software run and partly managed on their infrastructure. Several companies are also offering open source products in a similar vein. One step further still is the software as a service (SAAS) model, with the software only available through an end-user interface.

Both offer distinct advantages, including elimination of overhead and concerns about local management of hardware. Both options will also need additional considerations beyond functional requirements.

Cloud and hosted services for libraries often make use of central shared data stores or knowledge bases. An understanding of what can be stored and removed from such knowledge bases is essential.

Storage of personal details regarding staff and end-user information in a host or SaaS environment is also a major concern. Levels of uptime and service level agreements need to be examined. The level of access to the system (e.g. admin interface, configuration file, database connection) should always be sufficient for business requirements.

An accompanying tool has been created to assist in the procurement of SAAS.

Staff Access Privileges
Libraries serve users in many different roles and there can be a wide range of different types of people wishing to access information via any library system. It is important that users have the correct authorization to access the content so that sensitive data is not given to those that are not entitled to it. Meanwhile it is equally important that users who are entitled to see information are not denied access by the system

End User Access Management
Libraries provide content and services to end users, from book renewals to online journal access. In many cases, library software will sit in the middle between existing institutional directories of end-user identities and privileges (identity providers) and external resources subscribed to by the library. It will need to interact securely with both, ensuring that reliable access to third party services is provided to those that are entitled.

An understanding of the identity and rights environment in which software will operate can be vital in the selection of a product.

A Methodology for Software Selection in Libraries

3. Using This Tool

This decision support tool does not prescribe a one size fits all procedure for selecting software but rather aims to raise the relevant questions to be asked by libraries seeking to implement a new system. While the reader may be the project lead it is imperative that engagement with all potential users of the system is achieved as early as possible. For this reason this guide should not be seen as a tool for exclusive use by any one individual but rather as a reference for all those involved in the process.

This support tool is laid out below in sequential steps, however users may find it beneficial to jump over sections or even modify the methodology for their own purposes. There are two additional sections below focused on topics that should be considered before starting the process.

3.1 Managing Scope, Budget and Timing

As well as the costs and time required to implement the software solution you should also be aware of the investment required throughout the selection process. For those tasked with choosing a software solution there are normally a number of requirements.

These include:

Timescale

How long will it take to analyze and decide on the new software? Is there a deadline for this software to be in production? If so, it is important to plan out the time required using a Gantt chart or similar tool to determine the time needed for each step? For example, if you need to put proposals in front of a committee how much lead time do you need to set up a meeting?

Scope

There is normally no definitive answer when searching for the best solution. It is important to understand from the offset how detailed the analysis is expected to be. Is this a major system that requires detailed comparisons or a minor add-on that will only be used occasionally?

Management Reporting Needs

In addition to finding the best software solution for your library, there are commonly reporting or justification requirements. You will need to consider what evidence will be required to justify the decision you ultimately make and ensure that this data is collected throughout the process. Any examples of reporting from previous successful selection processes that can be shared will be useful.

Budget

How much time and resource is required or available for this process? Engagement is an important consideration but how many days input can you rely on from colleagues? Is there a budget to hire consultants? What percentage of your time can you devote to this process?

Especially when working in a large team it is important to ensure that relevant tasks are carried out at the suitable pay grade. More knowledgeable staff will be able to provide valuable input but their time will be more costly. Equally less experienced staff may spend an inordinate amount of time gathering information that more senior staff can readily supply.

3.2 Engagement

For any new software to be successful the people that use it have to feel comfortable with it and understand its purpose. It must also be closely tied to the operational strategy of your organization as a whole. Engagement with staff, users and other stakeholders is vital at all stages of the software selection process. One of the prime reasons for this is to mitigate risks by identifying as wide a range of use cases and scenarios as early as possible.

Engagement not only tends to increase positive perception of any new system once it is implemented but also allows risks and concerns to be identified before the system is rolled-out preventing costly adjustments at a later date.

Alignment with Institutional Requirements

Successful implementation projects must be closely aligned to the wider needs of the parent institution. In order to prove the value of investment and the effort and resource used during selection, the reasons for implementing must reflect the wider strategic needs of the greater body in which the library operates.

As an example, it may be as simple as proving a reduction in costs and streamlining of processes. Alternatively you may need to demonstrate how the new or replacement software will help the library better serve the needs of a specific user group. It may be vital in reaching a wider audience of strategic value to the institution or supporting a new initiative.

Where possible, direct links to high-level mission statements and strategic planning documents should be made in proposals, presentations and any reporting documentation. Such alignment is vital in winning support for a project from all levels.

High Level Support

It’s not a secret that successful projects often have some level of buy-in at the highest level. Ideally, a project sponsor with sufficient “clout” should be sought to protect and sell the procurement at the highest levels. Such support can be vital in ensuring budgetary and resource requirements are met and that administrative hurdles can be cleared in good time. They will also assist in maintaining parity between the implementation project and any high level strategic goals.

Internal Champions

If you are involved in a larger software selection project it is often worth assigning or recruiting internal champions that can push the agenda within their department or sub-department. From the start of the process it is important to reach out as widely as possible within your organization to get feedback from colleagues. This not only helps to create momentum but can also ensure management buy-in for any decision, as the decision will be supported across multiple parts of the organization.

Regular meetings either face-to-face or virtually are an important step in ensuring cohesion and making sure interested parties are in agreement over the needs of the software. You should aim to connect

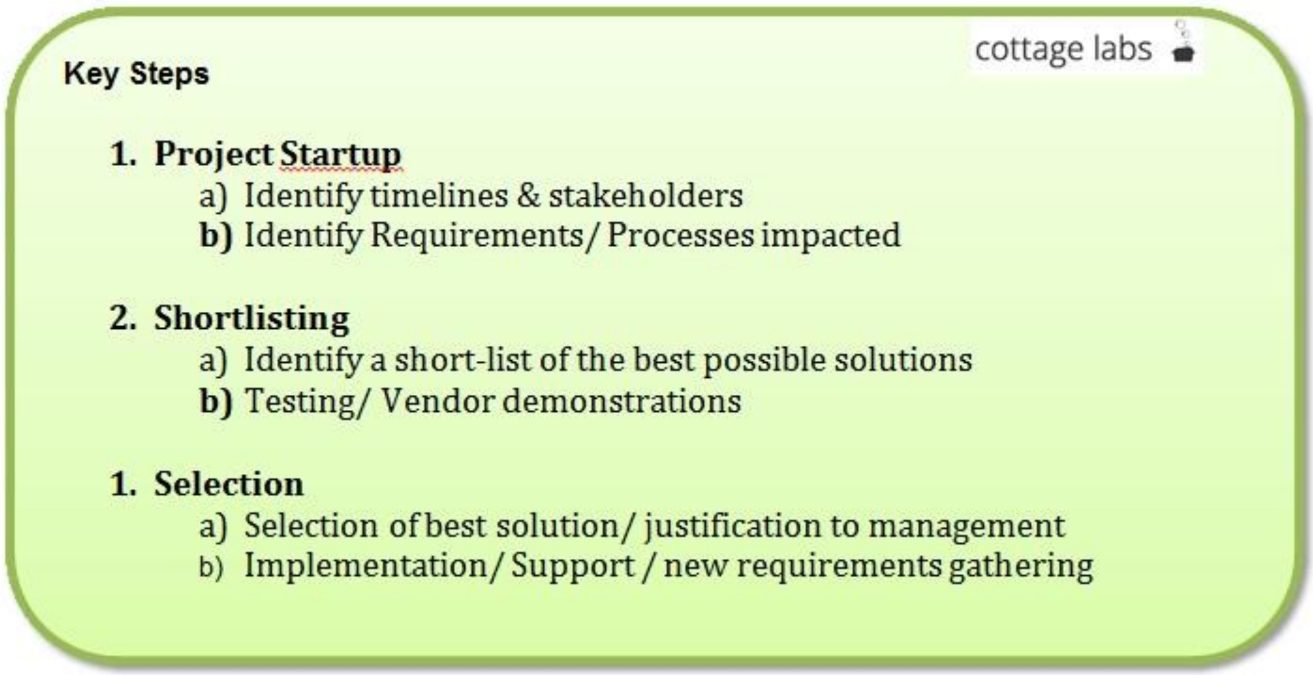
Iterative User Testing

With buy-in achieved at a number of levels, it can be capitalized on throughout selection by involvement in the testing of potential products. Internal champions and even friendly, interested end-users can all play a useful role in selection. If suppliers are unwilling or unable to make use demonstration software available, existing implementations at other sites can be used. Open Source software offers some specific advantages here, with the opportunity to set up bare bones local installations to view. Down the line, a group of end-user testers can be useful in implementation.

4. The Software Selection Process

The process of selecting software is never the same twice. Each step should be iterative, gathering input from all users and feeding this back into the decision process. As a result, the selection process should be intimately tied to the structure of your organization making it more than a case of simply ticking boxes.

With this in mind there are, nonetheless, certain broad steps are advisable to follow. These are listed below and described in more detail throughout this section.



cottage labs

Key Steps

1. **Project Startup**
   a) Identify timelines & stakeholders
   b) Identify Requirements/ Processes impacted

2. **Shortlisting**
   a) Identify a short-list of the best possible solutions
   b) Testing/ Vendor demonstrations

1. **Selection**
   a) Selection of best solution/ justification to management
   b) Implementation/ Support / new requirements gathering

4. Project Startup

4.1 Identify Timelines & Stakeholders

4.1.1 Timelines

The first step in any software selection project is to identify timelines. You need to understand what date the software needs to be ready to use. Will roll out of the software be at the same time across all areas of the library or will certain departments require it first? Will certain features be required before others?

Draw up a list of any hard dates that cannot be changed. Are any other departments depending on this software being ready by a particular date?

If you do not have any deadlines, estimate provisional timescales that you believe are achievable and make a note to revise them once you have more input.

4.1.2 Stakeholders

Once you have a rough idea of when the software will be needed, the next step is to identify the people that are likely to interact with it.

What is the Software Replacing?

New software may introduce a new set of processes or services to a library, or it may replace an existing set. This may be a like-for-like replacement, a replacement to a number of smaller tools or part of a larger integrated service. If this is the case, it’s vital to understand who makes use of the current tools, what they hate or love about them and what they would hope to gain from a replacement. Such individuals are likely to be key stakeholders.

Who Will Use the Software?

The range of people that seek services from a library is huge, and as a result the use cases for any software can be very diverse. The first step in identifying the user base is to outline the needs the software is likely to fulfill on a regular basis.

Key questions to consider:

- Will this software be integral to any job functions in your organization?
- What users will need the software to do their day-to-day job?
- Are there other users that will use the software less regularly?
- How often will they require access?
- Are these users library staff, part of a parent organization or members of the general public?
- Which users are you able to contact?

In addition to core users, it is worth considering people that are peripherally connected to the software but will not actually use it. For example, software may hold information about authors or staff even though they do not interact with the software themselves.

Who Will Operate the Software?
As well as those people that will rely on the software you will also need to consider who will be responsible for running it. There may be a variety of tasks associated with the successful upkeep of the software.

If it is a larger system that provides services across your organization these tasks may be split between several teams.

Who will be responsible for the following:

- Administrating the system, updating records or ensuring the information held is correct
- Customer support queries
- Applying updates to the software
- Adding new functionality
- Fixing outages or faults with the system

You may also need to check what resources will be allocated for these tasks. How much time do the people you have identified have to administer a new system? Are there other people that will be affected by the roll out or implementation of this new software?

Draw up a list of the most relevant stakeholders and get in touch with them as early as possible. Try to schedule a meeting with a cross-section of stakeholders to discuss requirements in more depth and identify representatives that can help in driving the selection process forward.

Who Will Pay for the Software?
The final, and perhaps most important, section of stakeholders are those who will be responsible for costs associated with implementation of the software. You should seek to clarify exactly what approval is needed and which individuals within your organization have a say in signing off purchases and covering ongoing costs.

It may be that you have a budget to spend and therefore do not need explicit approval; however you should still consider other contingent costs. For example, could there be any legal costs associated should the software fail?

Some relevant questions include:

- Who is responsible for costs? Will this be split across multiple departments?
- Which individuals will ultimately sign-off on purchases?
- What if costs grow?
- What are the requirements for cost justifications?
- Total Cost of Ownership (TCO) assessment?
- Return on Investment (ROI) assessment?
- Are there people within your organization capable of providing these assessments?

You should aim to speak to those individuals identified as early as possible and schedule time to understand their requirements from a strategy perspective. If the software is to be used across multiple departments, try to match their requirements as clearly as possible to specific functionality of the software. If any stakeholders require metrics, a business case or detailed assessments of costs make sure you have their needs in writing.

4.3 Identifying Requirements/ Business Processes Impacted

4.3.1 Licensing

Open Source vs. Proprietary

A key consideration when identifying requirements is the type of software license you will need. Open Source (OSS) offers more opportunities for customization but may require more development expertise within your library.

Some key licensing questions to consider include:

i. Does your library or parent organization favor a particular license model (OSS or proprietary solution)?
ii. Are you likely to make changes in future?
iii. What are the long-term costs for a proprietary solution?
iv. Ownership vs. licensing?
v. What extra resources are required to support an OSS effort?
vi. What is the rate of innovation for OSS vs. proprietary?
vii. Where do you expect to get support? If OSS, is there a well-established and active community?
viii. If OSS, is it supported by a commercial company?
ix. What are the risks of the software forking? Can you anticipate and manage this?
i. Are there any stability issues?
ii. If you do not have dedicated developers, is it possible to influence future development of the software or request new or specific functionalities?

4.2.2 Budget

Before you can make substantive decisions about which software you need you must first decide how much is available to spend on purchases. It is possible that the budget may be clearly defined from the outset but, in most cases, it will be contingent on other factors. It is important to determine what money is available as early as possible to guide the decision process. If it is your role to secure financing then you will also need to ensure buy-in.

In any larger software implementation there will be a variety of different costs such as upfront expenditure, capital costs, ongoing licenses, and ongoing operating costs. These may be met from different budgets, departments, other organizations or even contributions in different currencies.

Speak to all stakeholders that may contribute. Identify how much they can contribute and on what conditions. You should ensure that you have firm commitments, preferably in writing, from stakeholders that are responsible for contributing to the budget. If this is not possible at an early stage try to obtain an agreement on upper and lower bounds.

Some budget issues to consider include:

- What stakeholders can/will contribute to the budget?
- What is the approximate upfront budget available?
- What is the approximate ongoing annual budget available?
- Are there upper or lower limits imposed by any factors?
- How is the budget measured? Currency? Accounting?
- How does the budget time period fit with accounting year-end reporting periods?
- How do funds get released? Is this a capital purchase or expense? Who needs to sign off on purchases and at what stages?
- What Total Cost of Ownership (TCO) justification will you need to provide? Bear in mind that proprietary software is often not ‘owned’ but ‘leased’.

4.2.3 Business Processes

At its core, any software acquisition in a library aims to tackle a workflow issue by either providing new functionality for the end-user of library resources or improving efficiency. Understanding how the software will be used and where it fits within your library's workflow is vital to success.

Speak to the people that will carry out the relevant tasks at each stage of the workflow and write out step-by-step use cases. These will help you understand the processes that underlie each interaction with the software and how these may go wrong.

If you are replacing existing software, you must distill and preserve everything that is vital from the older processes in order to ensure continuity of business.

It may seem a time consuming step to map out business processes but the software selection period can be an excellent time to consider changes. You should be prepared to be flexible as implementing a new system is often the best time to remap workflows.

Where Will the Software Be Used?

Within a library there are normally a number of key workflows to consider when thinking about where the software will be employed. Depending on your library's organizational structure, these processes may be split across departments or teams in different ways. However, a general categorization common to most libraries would include:

- **Back office processes:** There are a variety of admin and back office processes that underpin the operation of any successful library. Software is widely used to automate resource procurement and description, or provide tighter integration between systems, or front of house processes. These processes include library workflow management, acquisitions, description, circulation, and inter library loan.

- **Front of house:** Front of house software tends to be aimed at library users. These systems are more likely to form the 'face' of the library; defining how outside users perceive the library's services. As a result usability issues are often more important for these workflows. These processes include discovery services, website CMS, booking systems, and self-service facilities.

- **Function specific:** - A third category of software is infrastructure. If it is functioning correctly it should be invisible to the user but nonetheless provide a vital layer of functionality: connecting services and ensuring compatibility and security. These processes include proxy server, Open URL, and identity management.

Once you have identified where the software will be used, you should ensure all major workflow requirements are mapped out with start points, key staff roles, end user profiles and required deliverables.

Remaining Flexible

If replacing existing software, a manual or paper-based routine it's vital that you remain flexible, rather than attempt to replicate the existing workflow with new software.

New software may provide new ways to approach a problem. It may simply render some aspects of the existing workflow obsolete by its very design. Attempting to mold software to exactly match an existing process is often a recipe for disaster.

Instead, identify the key deliverables from each workflow and ensure that they are met. Mapping workflows is often the best time to reflect on their value and focus on improvements, as well as ascertain the actual desired endpoint.

In many cases, the process itself may be obsolete. If you believe this to be the case, examine the end deliverable and the impact removing it from a library service might have. Usage statistics can be a vital tool here. Removing obsolete workflows that cannot be managed well by new products will greatly assist selection.

Back of House Example: Periodicals ‘Check In’

An integrated library management system (ILS) provides complex functionality to check-in periodical items. It can generate and maintain prediction patterns that automatically assign volume, and issue part metadata to new items when they are received. Although useful, such patterns are complex to configure and maintain as publishing patterns shift.

The ILS is approaching end of life with its software supplier and needs to be replaced.

An analysis of library usage data indicates that the shift to electronic subscriptions and open access publications over the past ten years has drastically reduced the need to maintain such data. As usage has continued to drop, 90% of all print journals have been moved to off-site storage. As such, complex metadata on exact holdings is no longer required and the functionality is not included in the required specifications. The workflow can be abandoned and staff time can be re-deployed to front of house work.

Front of House Example: Keyword-Based Discovery

The online public access catalogue (OPAC) function of an integrated library system (ILS) largely replicates a card index on the web. It requires users to preselect an index or combination of indexes to search (author, title, subject etc.) and enter terms accordingly. Many results screens involved browsing a list of ‘headings’ before seeing actual titles. The system is complex, full of library terminology and has a steep learning curve. It requires a high degree of user instruction to get the best result.

The ILS is approaching end of life with its software supplier and needs to be replaced.

An open source discovery platform allows library bibliographic metadata to be loaded into a modern XML database and fully indexed using the latest open source technologies.
Modern relevancy ranking can cope with simple keyword searches and return useful relevant results. Using post-search facets to limit results ensures that the libraries richly structured data is preserved. Thus, the end user search workflow is transformed into something quicker and easier. Readers are happier, and librarians spend less time instructing and helping users with online interfaces.

By considering front of house and back of house operations separately, the library is able to think about best of breed products based on specific needs, rather than as an ‘integrated’ solution that may not be a best match for either.

4.2.4 Functionality

With essential business processes determined and key deliverables established, required functionality needs to be described and rationalized in a codified fashion. This should be pitched at the right level for in-house staff, software suppliers and developers to understand. It should also be organized in a manner that could be matched to any scoring criteria.

Some specific elements will need to be drawn out. What should the software do? What should it not do? In some cases, it may be necessary to request a minimum attribute for performance (e.g. xxx connected users, xx requests per minute, scale to several million records, etc.)

Now is also the time to hone in on essential features and functions. Are there any features that are absolutely required? Again, consider the reasons for implementing new or replacement software and the greater strategic aims underpinning the selection.

Conversely, what would be a ‘showstopper’ if the system could not do it or it did not perform to specification? If a system cannot meet a vital requirement, it may no longer be worth considering as an option in the procurement process. Understanding what these points are at an early stage will again assist in procurement.

In all cases, denoting between ‘essential’ and ‘desirable’ criteria is a good methodology to follow.

(CONSIDER A SET OF KEY AREAS OF FUNCTIONALITY FOR MAJOR LIBRARY SOFTWARE TYPES/FUNCTIONS)

Learn from Others
All libraries have unique needs to match their user base, but they also share a lot in common. Existing descriptions of functional requirements for common software applications exist publicly (REFERENCE APPENDIX). Whereas sticking to them exactly is not recommended, they make excellent starting places for descriptions of required functionality, once required internal business processes and requirements have been recognized and understood.

Some Open Source software projects provide publicly available lists of functionality. This transparency is typical of the open source community. It can be useful in exploring the software as a viable option and in providing a benchmark for core functionality requirements.

Customization and System Access
Customization and control over software can vary widely in libraries. Some products have very limited options, and many commercial products restrict advanced customization options to the companies support staff only. Open Source products can of course be reworked from the low levels of code up, assuming you have the in-house skills to support the work.

If a product only needs to fulfill a specific need or has a few staff users, excessive customization may not be required. Extensive customization is best saved for end-user facing products.

All customization comes at a price, requiring a greater level of in-house support. It would be unrealistic to expect a third party vendor or community group to support your extensive interface reworking.

Some customization and system access points to consider:

General system configuration:

- How can you access and configure the system without editing code (admin interface, configuration file)?
- What skills are required (UNIX command line, XML etc.)?

User interfaces (UI):

- What level of interface customization is offered (configuring file, CSS+HTML headers, a template system, API access, source code access)?
- Who can customize the software UI? Staff? Supplier support only?
- What skills are required (knowledge of HTML, CSS, templating tools)?

Data loading/management:

- What options are available for batch loads and transformations of data (Batch change API, Graphical tool, write access to database)?

Reporting:

- How can you query the system to get reports on usage and activity (reporting interface, database access, third party reporting solution)?
- How well documented is the data structure underpinning the system?
- What reporting is vital to your business needs? Can any standard tools meet it?

4.2.5 Software Environment

No commercial or Open Source software can exist in isolation. In addition to describing functionality required from software, a description of the environment in which it is intended to operate will further assist suppliers and implementers in the selection processing.

Some key software environment issues to consider in a description:

- Network: Describe the network in which it will operate. Is there a closed safe firewall protected area? What level of network access will the system need to function effectively to reach end-users?
- Desktops: Describe the desktop environment in which a product will operate? Is client software required, can it update automatically?
- If a product offers all interfaces via a web browser, try to understand which browsers are supported. Is it locked into a specific OS/ Browser (i.e. Windows / Internet Explorer)?
- What operating system does the software require? Open Source products can often be recompiled to run on any operating system, to match in-house preferences.
- Can the software run on virtualized server infrastructure?
- What backup mechanisms and practices are supported or recommended by the supplier? How does this match against current institutional practices?

Software Architecture and Design

Any decision process should focus on business needs first. However, an understanding of the architecture and key components of the software is also very useful. Some idea of the technology behind a product may assist in avoiding a technological ‘dead-end’.

Some key software architecture questions to ask:

- Ask suppliers to explain which major programming languages are used. Are they covered by the skillset of any available in-house developers? Are the languages openly licensed?
- Understand which database and indexing software underpins the product. Is this an open source suite? Is there a choice of databases available?
- Understand any application frameworks or development tools used to create the product. Are these flexible and well supported? What license is attached to them?

Other points to consider:

- If the product is commercially licensed, consisting mainly of open source components, what is the additional value supplied by the developer?
- Is the product composed largely of licensed codebases and database software? If so, how much are they inflating the value of the product? Are they appropriate to business

needs given the size and scale of your library?

- Is the original product an open source one ‘forked’ away from the main trunk codebase by a specific vendor? If so, what were their reasons for doing so?
- How commonly used are the open source technologies in the product?

Understanding some the technological choices made in developing the product will give you some idea of what you are paying for, either in license fee or in developer time. This may also help you to in turn understand the ethos and attitude of the company or community behind the software.

4.2.6 Sustainability and Future requirements

Predicting the future is hard, but an understanding of potential institutional change within the lifespan of the product is helpful throughout the selection process and beyond.

Some future requirement issues to consider:

- How do you foresee the needs of your library changing? Is the system flexible enough to cope?
- What other software or systems is in your current roadmap, and will they be able to interact with this choice?
- Is your budget likely to go up or down? If you faced severe cuts what issues would this create with annual support or hosting costs?
- Is this product in a fast moving competitive market? If so, when would you plan to compare it against new and rival products?

Lifespan of the System

Given an understanding of future requirements, you will need to identify how long you wish to run the software before re-evaluating its use. Budgetary considerations are important here. The capital cost of any software license or major hardware purchase can also be written off over this period of time.

Choosing a longer-term lifespan may make internal operations easy to manage, as long as the system appears to be sufficiently flexible to change through that period. This will partly depend on the level of development from the supplier/community.

If things are moving faster, building a shorter lifespan into the software before reconsidering may be strategically useful down the line. Regardless of your choice, the system itself will have its own lifespan determined by the company or community that owns it. Some idea of this will affect your decision on when to review.

Upgrades

Commercial and open source products in active development will receive updates and upgrades at regular intervals. Some will be minor, others major. The speed and nature of such upgrades is often a key indicator of the level of interest in a product from its supplier or owning community.

- How often are you required to upgrade?
- How far behind can you track before supplier/community drops you?
- How are your requests for new features handled by the supplier/community?
- How are bugs found in upgrades handled?

Rate of Innovation

When comparing alternatives it is important to consider the lifecycle of different options. OSS often delivers innovation faster than proprietary software with more regular iterations and updates. Without the burden of a long product development cycle, open source software can create and release new features immediately by the people and organizations that need them.

Commercial Product Innovation

Understanding how a company approaches innovation can be difficult. Business practices vary and companies may not want to disclose information that may be seen as detrimental to competitiveness.

Some commercial software innovation issues to consider include:

- What % of profit is fed into development vs. sales?
- What % of workforce is engaged in development vs. sales
- What is the development cycle and roadmap for a product?

Open Source Innovation

Open source software has its own specific considerations. A strong supporting community is a must. Often, such communities are focused around one institution or a group centered on a funding source. Understanding the background to a product can be vital in gauging its usefulness to your organization.

Some Open source innovation issues to consider include:

- Is the code viewable in a public repository?
- How detailed is the code documentation? This can indicate the level of interest in development.
- Can you see the number of active developers? How are they concentrated, around a specific institution?
- Is there a product champion in the community?
- How is the community managed, are upgrades voted upon?
- What role could you or your institution play in the community?

4.2.7 Community and Support

Support

Once you have selected a solution there is the ongoing requirement to support the software. This might include changes to keep the software up to date or changes in your library's requirements that require additional modules. The type of software you choose will affect where and how you can seek support. One major difference in supporting software is that between Open Source solutions and commercial proprietary alternatives.

Commercial Support Considerations

This section applies equally to proprietary software and open source products supported by a vendor.

The company that created the software or a third party provider will generally provide support for commercial systems. Depending on the company this may mean that there is a dedicated support line that you can call to get support, however it will normally limit the options for your library to customize the software.

Some commercial support issues to consider:

- Understand what level of support is provided. What elements and problems are you expected to maintain in-house, what must you not touch?
- Understand the coverage of support. Is it 24x7 or 9 to 5, Monday to Friday? Greater support coverage will cost more and may not be necessary in all cases. Again, look back to your business processes.
- If you are unhappy with support, you need a clear defined path for escalation, including a named contact to take the matter forward.
- If possible, ask the company to outline the response plan for a downed system.

Many successful companies have user group communities based around their products. While independent of the vendor, they can be useful in raising common problems for all users and ensuring that escalated matters are dealt with. They also provide additional mechanisms to share ideas and experiences, and an alternative vector for support. When examining a user group, look at the organizational model, methods and frequency of communication and their relationship with the vendor.

Lastly, if you are sourcing support and development on an Open Source product from a company, consider their relationship with the wider community. Are they on the right committees? Is their code in the trunk or are they selling you a proprietary fork?

Community Support Models

Nearly all software is developed in collaboration with different developers contributing code. In the case of Open Source software, the community of developers can be extensive and include the majority of the software users as well as the original creators. The level of support can vary greatly depending on the particular project as some communities are more active than others. To determine if you can get the level of support required, you will need to assess the community around the project.

Some community support issues to consider:

- Does a particular solution have an active community ready to answer your questions?
- Determine what resources are available such as mailing list archives or forums. Try to connect with the community by posting a message.
- If you receive a reasonably prompt and helpful reply, this may be a sign that there is an active community of users out there ready to help.
- Good practice suggests that if you wish to avail yourself of such support, you should also be willing to provide support for other members of the community when you are able to do so.

Training Offered

Very complex systems may require specialized training from providers or other users. This applies to staff users as well as those responsible for running and configuring the system. Such training is likely to cost, in which case, this expense needs to be factored into implementation budgets. It should also be carried out at the right time in implementation.

Documentation

Complex software will also require good documentation. It’s important to consider documentation during selection, and request samples from commercial suppliers if necessary. Documentation should always be current and relevant to actual customer needs. Alternative sources of documentation should also be considered. Any public facing API’s offered by a system should also be documented in a useful fashion.

Open source products often rely on user communities to document products. This can form the basis of a wiki or public website. Being able to see such documentation can be very useful during a software selection process.

5. Shortlisting

5.1 Identifying a Short-List of the Best Possible Solutions

From the questions raised in the early stages of the selection process you should be able to identify factors that will be important to your selection. From your conversations with stakeholders you should hopefully identify several potential candidate solutions. You now need to decide on objective criteria to employ.

Criteria will not only help you to narrow your choices but will help to build a case for management and stakeholder buy-in. Decide which criteria can be measured and which criteria are more subjective.

You can use the OSS registry [link] to determine aspects of software that you want to consider or compare.

You should aim to write down as complete a list of potential solutions as possible, including software that may only partially meet requirements. Once you have this you can apply the criteria to each of the options. Try to rule out any solutions that clearly do not meet any of your requirements or are out of scope.

Some short-listing issues to consider:

1. What software has been identified by stakeholders in previous conversations?
2. Is there any software that partially meets my needs but that could be used in combination?
3. What other options exist to meet the particular needs of my library?
4. My library has developers with skills in specific technologies. Do they have preferences? What solutions mesh well with in-house skills? Where can my library go to get training, documentation, hosting, and/or contract software development for a specific open source package?
5. What are peers/colleagues in other libraries using? Can you speak to them directly?

5.2 Costs

At this stage it is worth revisiting your estimates for budget and costs. Now that you have more information on requirements and the processes involved you can assign more accurate costs.

For each of the solutions shortlisted assign costs for every stage

Hardware Costs

i. Purchase Price
ii. Maintenance

Direct Software Costs

i. Purchase Price
ii. Support and Maintenance

Indirect Software Costs

i. Administration of Licenses
ii. Transition Costs
iii. Staffing Costs
iv. Installation Costs
v. Training Costs
vi. Support Costs
vii. Downtime

5.3 Testing/ Vendor Demonstrations

Once you have ruled out any solutions that do not meet core requirements you should identify the methods of implementation and options to demo the software and RFP (Request For Proposal).

Will you be implementing this entirely in-house? If it is OSS can you find any examples of working implementations within the community? Will you require additional development, and if so who will do this? Will you require a third party/systems integrator? Will you be dealing directly with the manufacturer?

Bear in mind that there may be several different ways of implementing the same piece of software. These can have a profound effect on timescales, costs and applicability of the software. You should aim to summarize the degree of functional fit of the identified packages and list of possible gaps.

RFP

A Request for Proposal (RFP) is the process whereby suppliers are invited to submit a proposal to provide services or hardware. This process can be complicated and will generally require management to ensure that bids are compared equally and that requirements are met for a fair price. Many organizations have templates and practices in place. Check within your organization to find out if there are any guidelines that should be adhered to.

The first step is to check if your library or parent organization has any templates or practices already in place. If so, do you have to strictly follow them or are they only there as guidance? Most organizations will have some material based on previous RFPs; however this may not be coherent across different departments or may be out of date. If there are no examples from within your organization you can find templates online that may be useful as a starting point [link]. Always bear in mind that these should be considered as rough guides. Following a template created for a different scenario can lead to difficulties or unexpected costs.

A rough overview of what you should aim to include when creating an RFP:

- Background: Information about your library/organization
- Short project description: Stakeholders involved
- Necessary features of the software
- Any additional desired features of the software
- System integration requirements
- Preferences: License, technology, etc.
- Project budget
- Clear deliverables and timeframes/deadlines
- Contact information and deadline for submissions

6. Selection

6.1 Selection of Best Solution

Having considered all the relevant issues, the potential software choices should already be narrowed down. Of those remaining you should aim to identify any preferences from stakeholders.

There are a wide variety of approaches that prescribe a numerical or statistical approach to select the best software. These will typically require detailed analysis and measurement of key variables to which you can assign weightings depending on your requirements. A calculation is then made to compare which offering scores the highest.

At this point, you should decide whether a numerical/matrix approach works well for the sort of software acquisition you are involved with. While this approach is undoubtedly useful it must be carefully understood if it is to work, and in most cases requires a great deal of tweaking to successfully match requirements.

Some key questions to consider include:

1. Does a matrix/numerical-based approach suit my requirements?
2. What key factors must be given priority when applying weightings to variables?
3. Are there overarching strengths/weaknesses of particular solutions?

If you have selected a particular solution based on the questions identified and there is resistance from any parties try to understand where this is coming from. It may be that some stakeholders are resistant and may need additional convincing

By this stage you should have stakeholder buy-in and consensus from those relevant in the procurement process.

6.2 Justification to Management

Once you have decided on a solution, you should make sure you have the necessary collateral to justify your decision and push the acquisition through based on your organization’s approval process. Ask what justifications stakeholders require and seek to understand what format would best make the case.

Some management justification issues include:

- How does this purchase meet near-term goals or enable organizational growth or future needs?
- Is your goal in line with your company's immediate and short-term goals?
- Are there any cost savings associated with this purchase? How can you quantify or demonstrate these?
- Do you require information around Total Cost of Ownership (TCO)?
- Are you able to demonstrate Return on Investment (ROI)?
- Resolve resource restrictions.

6.3 Implementation/ Support / new requirements gathering

Following the selection of suitable software for your library the next step is to implement it and put it to use!

The selection process does not end there however...

Once your software is up and running you should gather as much information as possible about how it is used. This will not only be vital in assessing how well your new software is working but also help to shape future requirements specifications. Continue to gather feedback from the stakeholders you identified throughout the selection process.

No software purchase offers the ultimate solution. As technology changes you will undoubtedly have to choose new software or upgrades in the future. By measuring the impact of your newly implemented system, you will be better able to select software the next time around. You will also have better metrics to demonstrate value to stakeholders.

Appendix A - Other Methodologies

Since the early 1980's academic research has developed a series of mathematical models to compare selection processes.

Examples of these methodologies include:
The Adeli and Wilcoski (AW) Methodology
The Anderson (AND) Methodology
The Brownstein and Lerner (BL) Methodology
The USACERL (CERL) Methodology
The Edmonds and Urban (EU) Methodology
The Eskenasi (ESK) Methodology
The Kuan (KUAN) Methodology
The National Bureau of Standards (NBS) Methodology
The Subramanian and Gershon (SUB) Methodology
The Talley (TAL) Methodology
The Williams (WIL) Methodology
[Ref: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.4470&rep=rep1&type=pdf]

Open Source methodologies include:

- **Open Source Maturity Model (OSMM)** from Capgemini
- **Open Source Maturity Model (OSMM)** from Navica
- Methodology of **Qualification and Selection of Open Source Software (QSOS)**
- **Open Business Readiness Rating (Open BRR)**
- Open Business Quality Rating (Open BQR)
- QualiPSo **Open Source Maturity Model (OMM)**
- QualiPSo **Model** for **Open Source Software Trustworthiness** (MOSST)
- QualOSS Quality of Open Source

Comparison Chart (for OSS Methodologies)

Criteria	OSMM Capgemini	OSMM Navica	QSOS	OpenBRR	OMM
Seniority	2003	2004	2004	2005	2008
Original Authors/ Sponsors	Capgemini	Navicasoft	Atos Origin	Carnegie Mellon Silicon Valley, SpikeSource, O'Reilly, Intel	QualiPSo project, EU commission
License	Non-free license, but	Assessment models licensed	Methodology and assessments results licensed	Assessments results licensed	Creative Commons Attribution-

	authorised distribution	under the Academic Free License	under the GNU Free Documentation License	under a Creative Commons license	Share Alike 3.0 License
Assessment Model	Practical	Practical	Practical	Scientific	Scientific
Detail levels	2 axes on 2 levels	3 levels	3 levels or more (functional grids)	2 levels	3 levels
Predefined criteria	Yes	Yes	Yes	Yes	Yes
Technical/functional criteria	No	No	Yes	Yes	Yes
Scoring Model	Flexible	Flexible	Flexible	Strict	Flexible
Scoring scale by criterion	1 to 5	1 to 10	0 to 2	1 to 5	1 to 4
Iterative process	No	No	Yes	Yes	Yes
Criteria weighting	Yes	Yes	Yes	Yes	Yes
Comparison	Yes	No	Yes	No	No

Source: http://en.wikipedia.org/wiki/Open_source_software_assessment_methodologies

Features
Usability
Scalability
Security
Documentation
Reputation
Customization/Ongoing Effort
Upgrade Frequency/Versions
Stability/Version 1.0
Community Adoption Levels
System Support Needs
Features
Usability
Scalability

Grant Support Document
Features
Usability
Scalability
Documentation
Upgrade Frequency

Customization
Maintenance Requirements,
Community Adoption Levels,
System Support Needs,
Security

OSS Watch
Reputation
Ongoing Effort
Standards and Interoperability
Support (Community)
Support (Commercial)
Version
Version 1.0
Documentation
Skill Set
Project Development Model
License

Reputation
Does the software have a good reputation for performance and reliability? Word of mouth reports from people whose opinion you trust is often key. Some open source software has a very good reputation in the industry, such as Apache web server, GNU Compiler Collection (GCC), Linux, and Samba. You should be comparing *best in class* open source software against its proprietary peers. Discussing your plans with someone who has experience using open source software, and an awareness of the packages you are proposing, is vital..

Ongoing Effort
Is there clear evidence of an ongoing effort to develop the open source software you are considering? Has there been recent work to fix bugs and meet user needs? Active projects usually have regularly updated web pages and a development email lists of engaged users. They usually encourage the participation of those who use the software to provide feedback to aid in its further development. If everything is quiet on the development front, it might be that work has been suspended or even stopped.

Standards and Interoperability
Choose software which implements open standards. Interoperability with other software is an important way of getting more from your investment. Good software does not unnecessarily reinvent the wheel, or force you to learn new languages or complex data formats.

Support (Community)
Does the project have an active support community ready to answer your questions concerning deployment? Look at the project's mailing list archive, if available. If you post a message to the

list and receive a reasonably prompt and helpful reply, this may be a sign that there is an active community of users out there ready to help. Good practice suggests that if you wish to avail yourself of such support, you should also be willing to provide support for other members of the community when you are able to provide assistance.

Support (Commercial)
Third party commercial support is available from a diversity of companies, ranging from large corporations such as IBM and Sun Microsystems, to specialist open source organizations such as Red Hat and MySQL, to local firms and independent contractors.

Version
When was the last stable version of the software released? Virtually no software, proprietary or open source, is completely bug free. If there is an active development community, newly discovered bugs will be fixed and patches to the software or a new version will be released; for enterprise use, you need the most recent stable release of the software. There is, of course, always the option of fixing bugs yourself, since the source code of the software will be available to you. But that rather depends on your (or your team's) skill set and time commitments.

Version 1.0
In open source, there is no evidence as to the significance of a 1.0 version number. A program with a version number below 1.0 may be suitable for production use. Conversely, a product with a version number of 1.0 or above may not be. Criteria other than version number must be the guide here.

Documentation
Open source software projects may lag behind in their documentation for end-users, but they are typically very good with their development documentation. You should be able to trace a clear history of bug fixes, feature changes, etc.

Skill Set
Consider the skill set of yourself and your colleagues. Do you have the appropriate skills to deploy and maintain this software? If not, will you employ third party contractors or will you implement a training plan to match your skills to the task? Remember, this is not simply true for open source software, but also for proprietary software. These training costs should be included when comparing TCOs for different products.

Project Development Model
Open source development should not be chaotic, although it can sometimes look that way. An open source project should have a very clear development process that describes how contributions are made and how they are evaluated. It should also describe how contributors investing considerable resource in customizations can become a part of, or influence, the project management. This is to reassure significant contributors that their contributions will remain valuable to them in the future. For some projects, there is a formal structure governing

this kind of development, for others the structure is fluid. In both cases the rules of engagement need to be clear.

License

Arguably, open source software is as much about the license as it is about the development methodology. Read the license. Recognized open source licenses have well defined conditions for the contribution of your code to the ongoing development of the software, or the incorporation of the code into other packages. If you are not familiar with these licenses, or with the one used by the software you are considering, take the time to clarify conditions of use.

Bibliography

1. Bandor, M. S. (2006). Quantitative Methods for Software Selection and Evaluation. *Program*, (September).

2. Burton, A. (2006). Functional Checklist for Digital Repositories in the Research Quality Framework (RQF) Functional Checklist for Digital Repositories in the RQF Table of Contents. *Work*, 1-4.

3. Carpp, J. (n.d.). Software Selection Methodology. http://www.isaca-det.org/presentations/20090520_Software_Selection_Method.pdf

4. Fritz, C. A., & Carter, B. D. (1994). A Classification and Summary of Software Evaluation and Methodology, (940823). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.4470&rep=rep1&type=pdf

5. Method for Qualification and Selection of Open Source software (QSOS) GNU Free Documentation License. (2006). *Source*, (April). http://www.qsos.org/download/qsos-1.6-en.pdf

6. Müller, T. (2011). How to choose a free and open source integrated library system *OCLC Systems & Services*, *27*(1), 57-78. doi:10.1108/10650751111106573

7. Open Source Software Assessment Methodologies http://en.wikipedia.org/wiki/Open_source_software_assessment_methodologies

8. Top Tips For Selecting Open Source Software http://www.ukoln.ac.uk/qa-focus/documents/briefings/briefing-60/

9. Open Specifications for Library Systems http://libtechrfp.wikispaces.com/

10. Information to Include in RFP http://www.techsoup.org/learningcenter/techplan/page5507.cfm