Whitepaper: Selecting Database Software for Small Non-Profits

A Framework & Criteria for Comparison & Evaluation

With the inclusion of a number of new software-as-a-service offerings in recent years, the landscape of database options for non-profits has changed substantially. As a result, smaller organizations, ever-vigilant for opportunities to increase efficiency and improve capabilities, are considering the implementation of database systems in larger numbers. This paper reviews in detail seven major groups of decision criteria that drive the database selection process for a small non-profit organization.

Steve Hall

July 2nd, 2008

Contents

Whitepaper: Selecting Database Software for Small Non-Profits

A Framework & Criteria for Comparison & Evaluation

Introduction

This document reviews seven criteria groups that constitute a robust set of considerations for selection of database software. The criteria have been focused to address small non-profit considerations and have equal applicability to both community-service and faith-based organizations.

Why focus on *small* non-profit's you might ask? While non-profit organizations as a group generally have a different set of needs than those of their commercial counterparts, the size of the organization causes *significant* differences in criteria for software selection. This paper focuses on small non-profits as a distinct group, providing a relevant set of criteria for contrasting and comparing different approaches as well as different solutions.

Major Criteria Groups

As mentioned in the introduction, this paper organizes the selection criteria evaluation process into seven groups covering all aspects of the evaluation process. These groups are:

- Functionality
- Total Cost of Ownership (TCO)
- Usability
- Compatibility
- Reliability
- Supportability
- Extensibility

The rationale for this approach is derived from a similar methodology used by many product development organizations in the product development process. During examination of these seven criteria groups, in order to provide a framework for comparative analysis, this paper will contrast them within three different approaches to software acquisition:

- Build it in-house
- Purchase & run it in-house
- Software subscription service

Build it In-house

Most non-profits have access within their ranks of supporters to one or more technology professionals, or, at least a few technology enthusiasts that are capable of creating a hand-in-glove software solution for their organization. The price seems right (limited or no initial development cost) and the ability to build something that closely meets the organizations needs often make this an attractive solution. Also, with the advent of freely available open-

source software systems on which to build a solution, this option has become increasingly attainable.

Purchase & Run In-house

This is the traditional way that commercial software has been sold. The customer purchases a license to install and operate the software via an upfront fee and probably then pays a smaller amount for an annual support agreement. It is then the customer's responsibility to provide the infrastructure and operations processes for the software's on-going operation.

Software Subscription Service

With the advent of broadly available internet technology, the software industry has evolved its old "purchase and run in-house" model to a new and often more efficient model commonly called "Software-as-a-service" (SaaS), or, "Software Subscription Service". In this model, the software is managed directly by the vendor generally relieving the customer from any kind of infrastructure management responsibility and cost. This allows the customer to effectively outsource all maintenance and operational considerations to the subscription provider.

The following sections of this paper are organized around the seven criteria groups. Within the discussion of each group, we will examine and contrast the relative strengths of each of these three approaches to software implementation and look at the appropriate questions to be posed for each of these approaches.

FUNCTIONALITY

The first of our considerations and probably the most important is functionality. The fundamental question here is: "*Does it do what I need both now and for the foreseeable future*".

There are a number of dimensions to this question:

- Does it address process needs?
- Does it address information management needs?
- Does it incorporate suitable security features?
- Does it meet management and reporting needs?

Other related aspects such as accessibility and usability are covered elsewhere in this document.

Process Needs

This step may be as simple as preparing a list of the major processes that the application must address. If one or more processes are critical, complex, or specialized to the organization, it is useful to pay special attention to the inputs and outputs of the existing processes and compare them to the inputs and outputs of the processes supported by the new software to ensure compatibility. While the inputs and outputs do not necessarily need to match, the new inputs and outputs must make sense within the context of your requirements. In addition, it can be useful to determine:

- If existing data will be imported or uploaded
- If the software addresses probable future process requirements

Information Management Needs

In addition to process inputs and outputs, it is important to confirm that the application will accommodate all current data needs. It is useful to also ensure that process managers within the organization are empowered by the new software to implement additional data elements as needed without assistance from technical support personnel.

It is prudent to also ensure that all added data elements are included on reports and can be used to define filters both for reporting and demographic grouping purposes such as defining communication recipient groups if such functionality is supported by the software.

Security Features

Because of the heavy reliance on volunteer resources and the often sensitive information managed in a non-profit system, the security provided by the application is an important consideration. The security features should easily restrict both groups and individuals to specific areas of the system and also secure access to sensitive data fields as needed.

Management and Reporting Needs

In addition to standard reports that accompany all systems, the ability to clone and modify existing reports, and create new custom reports on an ad-hoc basis is an important feature. This will ensure that the system can meet specific organizational management information needs both now and in the future.

It also useful to ensure the database system will allow creation of exception reports based on any criteria. Exception reporting allows management to keep an eye on critical process metrics on an exception basis. To facilitate this, the database system should also support the automatic generation exception reports either when exceptions occur or on a scheduled basis.

Functionality versus Software Acquisition Models

Generally, the software acquisition approach will not have a substantial impact on functionality. However, it is worth mentioning that there *are* influencers that are generally inherent to each of the approaches:

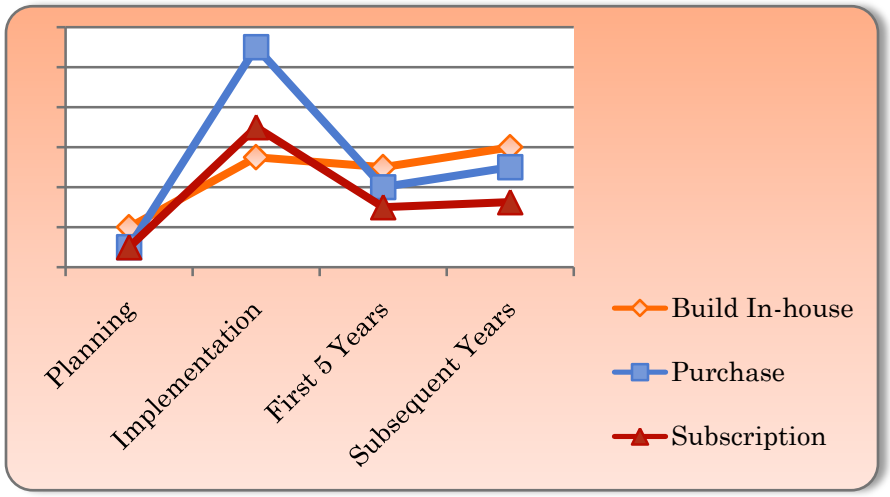
- The *build in-house* approach – when starting from scratch – tends to have less functionality due to the effort required to create, and implement the software as well as constraints of time, expertise and possibly budget. It does however enjoy the benefit of freedom to enhance and customize while resources remain available to do so. In order to mitigate the issue of reduced functionality, building on a base that uses one of the free open-source systems can give the development project a boost.
- The *purchase & implement in-house* approach tends to include more functionality due to the commercial nature of software having to meet a range of needs. An ongoing maintenance agreement will also generally provide upgrades to functionality over time. However, these may be infrequent and can incur additional cost.
- The *software subscription service* approach is the most likely option to produce frequent and inexpensive updates to functionality as the centralized and shared infrastructure nature of the product lends its-self to more frequent revisions.

Total Cost of Ownership

It is clear that the initial cost of software is only a small fraction of its ongoing lifetime cost. However, comparisons often do not include some of the hidden costs that can significantly impact smaller organizations.

Contrasting Three Approaches

Each of our three software acquisition approaches carry with them substantial differences in associated costs. While individual situations will have a large bearing on actual costs, the following chart depicts a cost profile that is characteristic of our three approaches.



Build In-house

While the volunteer-developed and/or free open-source in-house effort incurs the lowest implementation cost (costs only involve server, network and infrastructure-related investments), the expense of this option increases steadily over time.

The primary factors driving these expenses are the cost of on-going preventative infrastructure maintenance and support. This includes repair or replacement of computer hardware, networking equipment, operating system upgrades, and configuration changes.

It is also a well-known phenomenon that volunteer resources change over time. As a result, applications developed by volunteer resources are subject to significant future cost as those volunteers move on and become unavailable to support the existing application(s). This fact coupled with generally less than adequate documentation created for home grown systems greatly increase the risk and therefore costs of these implementations over time.

Many of the inherent costs associated with *build-your-own* systems can be mitigated by obtaining internet based open source software and hosting it externally as this all but eliminates the high infrastructure investments. However, the way an open-source system is implemented and how much the system is modified, can have a significant impact on future supportability and costs.

Recently, the writer encountered a small non-profit spending over $26,000 per annum to support a database and in-house email infrastructure that was implemented by volunteer

resources only 5 years earlier at a total implementation cost of less than $6,000. When this organization later migrated to a subscription service, functionality increased exponentially and costs reduced by about 75% after only a year of operation.

Purchase and Run In-house

In the case of purchasing software to run in-house, one would instinctively expect a larger up-front cost to implement and then lower ongoing costs each year. Intuitively, it would seem that the ongoing cost should be less than, for example, the software subscription model where the software is effectively leased as a service on a month by month basis. In reality, the comparison does not turn out this way for small non-profits as a result of the cost of infrastructure support.

While support contracts for the upfront purchase of software licenses can cost as little as one third of the rate charged for the subscription services, that cost plus the expense of maintaining the infrastructure (server hardware, operating system upgrades, anti-virus, patch management, network configuration, server and network management, professional support contract and more) can significantly exceed the cost of a subscription service. The dynamics of this cost comparison become clear once it is understood that with a subscription service, the infrastructure costs are spread across many clients using the service whereas with the in-house scenario, the cost burden for infrastructure is not shared.

Software Subscription Service

As has been noted, subscription services have the ability to provide software to meet the needs of the organization at a lower TOC than either built in-house or purchased software due to the savings on preventative infrastructure support, shared maintenance costs, and dedicated on-going resources to create software to meet both current and future needs. However, as the next section explains, not all subscription services are created equal.

Not All Subscription Services are Created Equal

It is important to further qualify any assessment by examining some of the different cost models used in subscription services available to small non-profits.

When purchasing a software subscription service, the areas of: concurrent verses named-user fees, additional module fees, and volume-based fees can add significantly to the cost of the software.

Concurrent verses Named-user Fees

The first major difference between services is the per-user licensing model. While most software is priced based on number of users, some pricing is based on *named users* while others are based on *concurrent users*. It is especially important for non-profits to look for the later model when there are a quite a few casual-use volunteers that will need database access. Implementing the named user model can require difficult tradeoffs to be made between user license numbers and the functionality available to volunteers.

Additional Module Fees

The second area where not all products compare equally is the pricing for additional functionality. While some vendors will supply additional functionality without additional fees, others will charge per module for each piece of added functionality.

Volume-based Fees

The third and final area of significant cost differential can be in the area of service volume pricing. Here, some vendors will charge a fee per item (per on-line credit card charge, per member record, per email sent, per event registration booked) while others deliver "all you can eat" for a fixed service fee.

In summary, when looking to select a subscription software vendor, in the area of pricing, always seek information on the three hidden cost areas – concurrent versus named user fees, additional module fees, and volume based fees.

USABILITY

An often overlooked aspect of comparative analysis is usability. Friendly, approachable, and easy to use are significant factors in the overall benefit that an organization derives from its software.

The User Interface – Three Basic Approaches

The great majority of the newer "internet based" software works from within a web browser whereas historically, most software was "installed on your computer" and provided a much richer interactive experience than that provided with a web browser. Generally, when presented with the option of performing the same task using a web browser or an installed application, most users will gravitate towards using the installed application.

This difference in preference is due to a number of factors: responsiveness, number of "clicks" necessary to complete a task, the "flat browser" surface compared to the "thick and rich" application surface, and the ability to perform a number of tasks concurrently compared to a single task at a time.

Recently, a number of vendors in the leading-edge internet-enabled constituent relationship management (CRM) space have realized that the older "*rich installed application*" interfaces are generally preferred to the two dimensional *browser based interfaces* and hence, have started investing in a third, hybrid approach – "*rich installed internet enabled applications*". It is useful to understand which model is used by each software application and carefully weigh the relative merits with respect to usability.

Intuitive Design

While a friendly and capable rich user interface is significant all by itself, an intuitive design is an equally important aspect of software performance.

Friendly approachable software that intuitively works the way one would expect makes it easy for staff and volunteers to complete tasks and supports process conformance. This

improves organizational efficiency and quality as users are willing to use the software to manage designated processes. Difficult to learn software encourages users to find alternate paths and workarounds to get their work done, ultimately leading to process divergence, reduced efficiency, and a reduced quality of output for the organization.

Given the difficulty of measuring usability, this aspect of the software selection process is often not given the weight it deserves. While one may not necessarily feel comfortable comparing software by using a non-scientific qualitative score for "intuitive usability", this paper recommends that you do exactly that. Take a staff person or volunteer that will be responsible for using some of the basic system functions and have them attempt to use the software without any instruction. They will probably turn down a few dead ends and head down some one-way streets the wrong way, however, intuitive software should help them ultimately complete the simplest of tasks without the need for training.

Complexity & Feature Bloat

Another significant contributor to usability can be the complexity or "feature bloat" of the software. Complexity is often caused by the software displaying far more functionality than is needed by any one particular user. Dealing with additional fields, screens, and unfamiliar terminology can cause users to view the software as too complex, whether or not they are required to actually *use* those areas of the software.

Because functionality required by one user can be viewed by another as unnecessary, removing the functionality is often not the best approach. The complexity issue may be better addressed by ensuring that users do not encounter fields, screens, and terminology that is not applicable to their particular function in the organization. This can be achieved in some software through user-based configuration that hides elements of the application not applicable to a user's function.

Training Resources

Due to the nature of non-profits and the likelihood of high levels of turnover in volunteer staff, it is useful to select a software product that includes video and self-paced instruction modules in addition to reference documentation and/or classroom and on-site training services.

Video and self-paced instruction, delivered via the internet will positively impact the organization not only by reducing training costs but also by ensuring that users are more effective as a result of their improved knowledge of software operation and their ability to accurately perform processes managed by the software.

Accessibility

Another trademark feature of the Subscription Service model delivered via the internet is its inherent ability to make information available anywhere, any time. This ability is further augmented by the option to retrieve information via any internet-connected device (such as a phone) using a web browser, assuming such devices are supported by the software.

Compatibility

In today's connected world, islands of information can easily lead to process inefficiencies and data inaccuracy. Such data island issues can be avoided by incorporating data compatibility criteria into the application planning and acquisition process.

At a basic level, data compatibility should include the importing, exporting and merging of data between the database application and productivity tools such as Microsoft Office®.

Exporting *from* the database to productivity applications supports activities such as mail merge, analysis, tabular and graphical reporting, electronic communications, and presentation content. Importing *to* the database provides for adding to database records with supplementary information from spreadsheets and other data sources. Such data can reflect real world information (event attendance, activities, history, data from other systems) that is useful when combined with data in the organization's primary database.

An organization that has implemented two or more separate database applications such as CRM, email campaign management, donor management and wealth screening will need to contend with import/export issues to keep all data synchronized. In this instance, compatibility of import/export formats between applications is essential as it is unlikely that smaller non-profit organizations will find automated data interfaces an affordable alternative. If however, the organization employs a fully integrated database application, it will not need to contend with import/export issues due to the integrated nature of the system.

Reliability

Reliability, generally measured in "uptime", is a factor of application performance and dependability as well as the performance and dependability of the infrastructure supporting the application.

When critical applications are run on internal infrastructure (as in the case of *built in-house* and *purchase & run in-house* scenarios), the reliability and performance of the infrastructure – always important – now becomes as critical as the availability of the software running on it. This fact generally requires an organization to introduce a degree of redundancy in its infrastructure by adding backup servers, redundant disks, off-site rotational backups, physical security against theft or destruction, and other measures. While these measures themselves incur additional cost, the majority of the cost burden can come from the need to manage the added complexity in the infrastructure and introduce risk mitigation strategies through addition of technical maintenance and support services.

When utilizing the *subscription services* scenario, the focus on reliability changes. In this scenario, the most significant aspects of the infrastructure and management responsibility for it are outsourced to the service provider. In this instance, it is important to ensure that

the service provider has incorporated reliability and redundancy measures such as failover servers, redundant disk arrays, multiple network paths and data center security. From a cost management point of view, this approach makes sense as it shares the cost burden of these measures across the many organizations utilizing the provider's shared infrastructure.

If you have read this section of the paper and feel overwhelmed with the potential complexity and management responsibility of the items addressed, then the subscription service route is probably the option of choice for your organization.

SUPPORTABILITY

Supportability addresses three distinct aspects of ongoing operational management:

- Remedial support services
- User support services
- Application design for supportability.

Remedial Support Services

There is no good time for something to go wrong with application software. Performance issues, broken functionality, or access issues will need end-to-end technical support services to ensure that the problem can be identified and rectified.

In the case of *built in-house*, addressing support considerations can be complex. If the database system was created by volunteers, it is important to ensure that appropriate troubleshooting and remedial support documentation has been made available. Combined with the documentation, a contracted support services company retained for such support emergencies should be able to return systems operation to a nominal state with minimal interruption to operations. Such an arrangement is prudent because it removes the reliance on immediate availability of the volunteer resources to rectify system operational issues. While the volunteer may be a long standing member of the non-profit community and there is a high level of trust in the individual concerned, failures will eventually occur when the volunteer is unavailable for an extended period of time or not at all.

In the case of *purchase & run in-house,* the software supplier will generally provide the option for ongoing support and maintenance services. These services may or may not include the option to support hardware and operating systems on which the application runs, so it is important to determine if there are gaps in the support contract that will need to be addressed. In this case, you will also want to ensure that there is an agreed protocol for either a lead vendor, or single point of contact for problem ownership built into your contracted support services. This added precaution avoids the issue of finger pointing when those stickier issues make it unclear where the problem really lies.

In the case of *subscription services*, remedial support is included as a key aspect of the service delivery and should be addressed in the agreement with your service provider(s). This agreement will generally include a single metric expressed as "uptime availability percent." In this scenario, the most significant part of infrastructure and service delivery support is handled by one or more vendors – the subscription service provider(s).

Even with outsourced remedial support, there are still a few minor support considerations to be addressed such as local network and desktop infrastructure support – items which are probably covered by existing arrangements. It is prudent to ensure that these existing support providers are familiarized with the new subscription service(s) so as to be in a position to contribute to remedial support activities when needed.

In the event that multiple external subscription services are used to meet overall database needs, it is important to understand where to turn for assistance with data exchange and interoperability between the multiple services. If a *fully integrated* service is used, support for data exchange and interoperability will not be necessary as functionality will already be integrated within the one service.

User Support Services

More often than not, support needs will be related to users not knowing how to complete a specific task. This is common in the initial months of using a new system, so it is important to ensure ready access to live support services in the early months of implementation.

Good documentation and task-indexed on-line video walk-throughs are very effective for ongoing task-based assistance. In the event that these tools are not available or an answer cannot be found using them, timely access to support professionals from the subscription service vendor is essential. If such add-on services are charged beyond a certain level, it can be prudent to ensure that there are processes in place to vet all requests from your users prior to them being actioned by the vendor. Some vendors provide a service that allows you to triage incoming requests as part of their support ticket management process.

EXTENSIBILITY

The last set of criteria to be considered is in the area of extensibility. While the current needs for a system are probably well understood, needs do evolve over time. Changes needed in the future may be small like needing to capture additional information for a process or may be major such as adding e-commerce functionality to the web site.

The Data Integration Challenge

It is always possible to add an additional piece of software to address a new requirement. It is important to keep in mind that doing so may create an additional "data island" which can create future data integration issues.

Multiple database applications (CRM, e-Commerce, Customer & Inventory Management, Donor Management, Electronic Communications Management, Inquiry tracking and other process based systems) will at some point cause the organization to address data duplication and contention issues. Problems like "which of the multiple addresses and contact information for these supporters is correct" will crop up time after time. Processes must be created and managed to address the complexity of these issues.

It is easy to see that integration across all of these data management areas is a most useful requirement. As a result, when selecting an initial system, it is useful to plan to meet all of the organization's immediate needs by ensuring that all applicable areas are addressed by a single application where possible.

Extensibility through Modules

The aspect most often *not* considered at the time of system selection is planning for the future unknown needs of the organization. The simplest method for accommodating these unknown needs is ensuring that the initial system is designed to be extensible. This means having the ability to integrate additional modules of functionality, both standard and custom.

Additional modules are generally made available by either the vendor supplying the system or by an ecosystem of partner suppliers. In the later case, the services of a consultant are usually needed to manage the complexities of integrating the additional modules and to manage the ongoing complexities of the integration. It is important to factor in consulting costs over the long term for building such a multi-vendor integrated solution.

Extensibility through Open API's

The second aspect of extensibility is more complex and involves utilizing open, published interfaces that support interoperability and data integration with external applications. These interfaces, generally called application programming interfaces or API's, are generally implemented with copious amounts of documentation that explain to the programmer and other development personnel how they can be used.

In most cases, the nature of this type of extensibility restricts its usefulness to larger non-profit organizations that have internal IT staff and budgets that accommodate specialized development projects.

With the advent of subscription-based software services, these API's are implemented using a relatively new technology called Web Services. While still technical in nature, these web services do make for a less complicated approach to creating custom applications to deal with the special needs of an organization. Consequently, it can be useful to at least ensure that subscription-based services have open, documented web service interfaces available should the need for custom development projects arise in the future.

Resources

The criteria detailed in this whitepaper are available as a software comparison worksheet to assist in the evaluation of software packages for a small non-profit organization. The worksheet (an Excel worksheet file) and an electronic version of this document is available for download from:

http://www.connect4.net/Site/DBSelectWhitepaper.aspx

About the Author

Steve Hall has spent over twenty five years in the information technology industry, developing software and managing infrastructure. He has founded two database solution startups and spent almost ten years managing international technology infrastructure for Hewlett-Packard Co in Asia Pacific. Having more recently completed a three year stint as Executive Director for a Silicon Valley based non-profit, Steve started Connect4 where he now serves as CEO. Connect4 is the supplier of Gnosis – a new, integrated software-as-a-service database solution for small to medium sized non-profits.

Copyright