MULTIMEDIA TRAINING KIT

CHOOSING OPEN SOURCE SOFTWARE HANDOUT

Developed by: Mark Surman and Jason Diceman, The Commons Group, for the Association for Progressive Communications (APC)

Table of Contents

About this document

These materials are part of the Multimedia Training Kit (MMTK). The MMTK provides an integrated set of multimedia training materials and resources to support community media, community multimedia centres, telecentres, and other initiatives using information and communications technologies (ICTs) to empower communities and support development work.

Copyright information

Introduction

Whether you are working with open source or commercial tools, picking the right software can be a difficult process. Often, we don't discover that we have made a software choice until it is too late – when we are already using the software in our organization. Also, it is easy to be swayed by an attractive new feature or compelling promotional language on a web site, even if the software in question has certain deficiencies.

The best way to avoid these pitfalls is to invest in a thorough and thoughtful review process before choosing a new piece of software. The simple **3 step method for open source decision making** outlined below is designed to guide organizations through exactly this kind of process. Considering both organizational needs and technical issues, this method can be used by any one with basic information technology and organizational planning skills.

This process works well for all kinds of software – server, desktop, web applications. However, the degree of detail and analysis required will vary depending on the situation and type of software being reviewed. For example, the process of selecting a file server platform to support a single work group can be done very quickly and with very little testing. In contrast, choosing a new word processing application or e-mail client for hundreds of different people in multiple locations will require a more rigorous process with extensive testing. Also, projects

that require software customization – especially web application projects – require additional planning related to the programming and development process.

Step One: Define your needs and constraints

The first step in the process is to clearly define your needs. This should include both the overall needs of your organization as well as the needs of individual users. Specific issues to consider include:

- **Organizational needs** – What problem are you trying to solve? Why is your organization seeking new software?
- **User needs** – What do individual users need to be able to do with the software? Are there particular things users have already asked for?
- **Features** – What are the actual features that must be provided by the software? How important are each of these features?
- **Language** – What languages does the software need to accommodate?

At the same time, it is also important to consider the constraints that your organization is under in considering software. Issues to consider include:

- **Budget** – While the software may be free, there will definitely be training and integration costs. How much do you have to spend?
- **Timeframe** – How quickly do you need to implement the software? Does it need to be something that can be up and running tomorrow? Or is there time for customization and configuration work?
- **Compatibility** – Are there legacy systems that the new software must work with? Does it need to run on a particular platform?
- **Skills** – What skills do your existing information technology staff or volunteers have? What skills do end users have? How adaptable are people to new software?

All of these factors will play a major role in the process of identifying your software options and making a final software selection. Given this, is important to write down information about all of these factors.

Step Two: Identify your options

The next step in the process is to come up with a short list of three to five software packages that are likely to meet your needs. This is basically a process of reading through information about various software packages and comparing them against the needs and constraints you listed in the previous phase. There are a number of places that you can look for open source packages to review:

- **Recommendations** – Ask people you know what packages they have used and liked in the past. These people could be from other NGOs or from organizations that provide technical support to NGOs.
- **This guide** – The list of additional resources which accompanies this guide includes a list of mature open source software options in a number of categories. This is a good starting point.
- **Reviews and directories** – There are a number of good open source directories and review web sites. Good places to start include OSDir.com and OpenSourceCMS.com.

- **Software package sites** – Most open source software packages have their own web site. These sites usually contain promotional information and documentation that will help with your review.

Using directories, reviews and software package sites, you should be able to determine which packages are able to meet your basic requirements. When you find a package that seems to fit the bill, you should compare it against the detailed needs and constraints list using the **Software Package Review Worksheet**. If a package meets most or all of your needs and constraints, you should add it to the short list for detailed review.

Step Three: Undertake a detailed review

Once you have identified your options, you are ready for the final step – reviewing and choosing a software package from your short list. At this stage, all of the packages that you are reviewing should be generally suitable for your task. The aim of this section is to assess which of the possible options will be best for your organization. This assessment can be done by rating each package against the following criteria:

- **Quality** – How well do the features you need seem to work? Do you like how they have been implemented?

- **Ease of use** – Is the process of using the software intuitive and obvious given the skills of the people who will be using the software? Or is there a steep learning curve?

- **Ease of migration** – If moving from another software package, how hard is the migration process? Is it likely that users will have a difficult time adapting?

- **Stability** – Does the software crash often? Is a lot of effort required to maintain it and keep it running?

- **Compatibility** – Does the software use file formats and communications protocols that are based on widely accepted open standards? Is it compatible with other systems you are using?

- **Flexibility** – How hard is it to customize and adapt the software to your organization's needs? Will the software grow with your needs? Is it scalable?

- **User response** – When given a chance to test the software, how did users respond? Were they able to figure it out? Were they excited about the way the software worked?

- **Buy-in** – Is there broad support for a particular package within your organization? Are there any major detractors? Active support or resistance for a package can have a major impact on successful implementation.

- **Wide use** – Is there evidence that others are using this software package? Does the popularity of the package seem to be increasing or declining?

- **Support community** – Is there an active online support community? Are there recent postings to the support mailing list? If you post a question, does someone from the community get back to you with a helpful response?

In ranking software against these criteria, hands-on testing is the key. Each piece of software should be installed and tested for quality, stability and compatibility. A group of key users should also be given the chance to try out the software in order to assess factors such as ease of use, ease of migration and user response. Information about usage and support can be gathered by looking at the software package's web site. If the support forums on the site are not active, it is unlikely that the software is widely used or that support will be available.

In terms of ranking, each software package on the short list should be rated against each of the criteria above. A score from one (insufficient) to five (excellent) should be given for each criteria. Ideally a short note on the rationale for each score should also be included. This

information can be collected using the **Software Package Review Worksheet**. Scores can be compared using the **Software Comparison Worksheet**.

Making a decision

Once you have completed the 3 step method, you should have a score for each of the software packages on your short list. This score should be a good indicator of which package is best for you. If a package scores very low, it is likely that it will cause problems once implemented and should not be used. This said, you should also use your intuition. If two packages are close in score, your gut feeling about the "right" package is probably more important than the actual numbers.

Looking for a second opinion? *Check out David Wheeler's "How to Evaluate Open Source Software / Free Software (OSS/FS) Programs" document. Wheeler offers a methodology similar to the one offered here, but more from the perspective of a hands-on technical person. The document can be found at: http://www.dwheeler.com/oss_fs_eval.html*

Will it save us money?

One of the big questions with any software decision is "will it save me money"? Answering this question properly is best done using a "total cost of ownership" (TCO) approach. This means considering all of the different costs that will be incurred over the lifetime of a particular piece of technology – hardware, software, maintenance, training, programming, testing, upgrades. Without all of this information, it is impossible to really know which software solutions are going to be the most cost effective.

While there are great debates on the topic, there is a great deal of evidence that mature open source applications offer a lower total cost of ownership than their commercial counterparts. In an article entitled “Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!”, David Wheeler lists the main reasons why open source comes out cheaper:

- **Open source costs less to initially acquire** because there are no license fees;
- **Upgrade and maintenance costs are typically far less** due to improved stability and security;
- **Open source software can often use older hardware** more efficiently than proprietary systems, yielding smaller hardware costs and sometimes eliminating the need for new hardware;
- **Experience shows that open source is cheaper** especially in server environments, with many case studies now demonstrating lower TCO for open source.

Wheeler's article also includes many detailed examples and links that show how open source has saved money in particular circumstances. See: http://www.dwheeler.com/oss_fs_why.html for these details.

For more information on calculating total cost of ownership, you may want to visit TechSoup (see: http://www.techsoup.org/howto/articlepage.cfm?ArticleId=295) or the Council on School Networking (see: http://classroomtco.cosn.org/gartner_intro.html)

Beyond decision-making: migration and training

How will we make the switch?

If you are planning to use open source software to replace an existing system, you will need to deal with the question of "migration". Migrating from one platform to another should be handled using a careful and phased approach. The European Union has published a document entitled the "IDA Open Source Migration Guidelines" that provides detailed

suggestions on how to approach migration. The document starts with the following recommendations:

- Before starting have a clear understanding of the reasons to migrate;
- Ensure that there is active support for the change from information technology staff and users;
- Make sure that there is a champion for change - the higher up in the organization the better;
- Build up expertise and relationships with the open source movement;
- Start with non critical systems;
- Ensure that each step in the migration is manageable.

The IDA guide is an excellent place to turn for information before starting the migration process. It includes information about leading open source applications in a variety of categories as well as detailed scenarios describing migrations from common platforms (e.g. an all Windows desktop and server environment). The guide is online at: http://europa.eu.int/ISPO/ida/export/files/en/1618.pdf

How should we approach training?

The other issue that needs to be seriously considered is training. Getting the most out of any technology requires investment in both formal and informal learning for users. Once the migration and testing phases are done, consider:

- **Formal training workshops** for all of people who will be using the software you have put in place. This is especially important for non-technical users who may not be used to learning new tools on their own.
- **Informal peer learning sessions and networks** that encourage users to help each other. Many people learn most effectively from knowledgeable colleagues than they do from a "teacher". Consider ways that users can help each other learn outside of the classroom setting.
- **Printed "cheat sheets"** that provide basic information about how the software works and can be useful in the context of your organization. This approach is very useful if you are using open source to replace an existing tool as the cheat sheets can highlight and explain features that have changed.
- **A list of links to online support resources** that relate to the software you have put in place. Each link should have a short description explaining the kind of help it can provide.
- **Information about the open source community** that supports the software you have chosen. At a minimum, this should include web site links and mailing list sign up instructions. This information will be very valuable for system administrators and developers in your organization.

The exact mix of approaches that make sense for your organization will depend on your circumstances. Installing a new office suite on everyone's desktop probably will require some kind of training.