BUYING A DEVELOPMENT SYSTEM: AVOIDING FASHION MISTAKES

Robert L. Weiner
President, Robert L. Weiner Consulting
robert@rlweiner.com
www.rlweiner.com

Software selection is like shopping for clothes. You can buy something you'll love for years, or something you'll be ashamed to admit owning one month from now.

Technicians often select alumni/development systems, since it is their job to develop applications and keep them running. But buying software is a critical business decision for you in the alumni or development office. While technical considerations are crucial, the system must also meet your business requirements. This chapter discusses the major steps involved in selecting packaged software.

BEFORE YOU START: DECIDE HOW YOU WILL DECIDE

Who will make the selection — one person or a team? Technical staff or users? If you include both groups, whose needs take precedence? Will this be a consensus decision? Will you issue a Request for Proposals (RFP)? Who will be in charge of managing the process?

A decision document that outlines how you intend to resolve differences helps to build consensus and can be a lifesaver later if problems arise. The document should include the roles and responsibilities of decision-makers, their limits of authority, and the process to be used if appeals are needed. Agreeing to these rules before you begin will help you avoid charges of "stonewalling" or "railroading" later if factions develop in the selection process.

PLAN AHEAD FOR BUY-IN

If you get buy-in from users during the selection process, they're more likely to support you during the implementation. If stakeholders believe a software package was forced on them, it's difficult to change their minds later. This is not to say that software selection is a consensus decision. But users need to understand how the system was selected and that their interests were taken into account. If you don't get buy-in from the start, all subsequent decisions are likely to be questioned. Be consistent in your communications to interested parties; if your process is honest, fair, and reasonable, you should find support.

ASSEMBLE THE TEAM

Assuming that you're using a selection team, who will be on it? Who will be in charge? What constraints will they work under? Does anyone else need to approve the decision? Will they need any assistance?

Selection teams usually have at least three members and no more than 12. While the team should be broadly representative of the user community, it needs to be small enough to work efficiently.

The team's role will be to understand thoroughly your requirements (and the RFP if you use one), understand your selection criteria and rating and weighting schemes (if you use them), read and rank any RFP responses, and attend and rate the product demonstrations. Team members may be required to visit institutions that use your selected system. This will require a significant time commitment.

One member of the team should be the project leader. The leader is responsible for keeping the team on track, making sure they have a consistent understanding of the selection criteria and process, taking care of logistical details, checking references on the finalists, and addressing any questions or problems. The project leader will also play a key role in communicating the status of the selection process to interested parties.

DEVELOP DETAILED REQUIREMENTS: WHAT ARE YOU LOOKING FOR?

Why are you looking for a new system? What's going to make it better than what you're using today? The first step in the selection process should be an assessment of your needs.

The needs assessment should start with the "big picture." The software should allow you to achieve your strategic vision of how your operation should run and where it needs to go.

The needs assessment can be the most time-consuming part of the selection process, particularly at a large university. Obviously, you'll want to meet the needs of the units in your office. And you might want to establish "best practice" goals based on what other development operations are doing. You might also need to consult with other departments (such as finance, registrar, internal audit, and the computer center) to determine the full set of requirements.

At the end of this process you should have a list of functional and technical requirements for the new system. The requirements document can easily be turned into an RFP, and can be used as a scorecard for rating systems.

KNOW YOUR LIMITS

Your selection might be limited by your budget, the time available for implementation, your staffing resources or technical talent, required or prohibited technologies, or your need to interface with other systems. You need to answer questions such as:

- Are you looking for a “best of breed” solution or are you buying into a campus-wide, integrated system?
- Will you pay to customize the system to meet unique requirements or should the system meet your needs without modifications?
- Does your fund-raising strategy require leading-edge technologies?
- Are you limited to a system that runs on specific computer hardware or operating systems?
- Does the software have to run on a specific database management system?
- Does the system have to work with both Macintoshes and PCs?
- Does it have to have a graphical or Web interface?
- Does it have to be a client/server system?

The selection team needs to know (or determine) these constraints.

HOW DO YOU RATE?

No system is going to do everything you need, so you need to know which requirements are most important. One way of doing this is to document mandatory requirements and screen out any system that can’t meet them. You can also assign points to each requirement based on how critical it is in meeting your goals for the system. You can use a simple point system (i.e., 15 points for mandatory, 10 points for “nice to have” and five points for “wish list” items). You can give extra points for features that aren’t mandatory but are important for your long-term vision of how you’ll operate.

You might assign a weighting factor to an entire category (i.e., technical requirements equal 20 percent of the total score, donor management/gift processing features equal 25 percent, alumni/membership features equal 25 percent, references equal 30 percent, etc.). You can also create ratings for the demonstrations, particularly if you’re asking all the vendors to demonstrate certain features or scenarios. Ratings and weightings help take the subjectivity out of evaluating RFP responses and demos.

Be cautious about designating mandatory requirements, however. If you use the term too loosely, you might inadvertently exclude most or even all potential vendors. But if you don’t use it enough you might wind up with a system that lacks critical features. And remember your strategic goals. Mandatory requirements are often purely technical; don’t lose sight of your business needs amid the sea of technical details.

It’s often helpful to place a limit on the total points that can be assigned to your set of requirements. For instance, if you have 50 requirements you might limit total points to 400 to prevent everything from being assigned 15 points. You might even limit the total mandatory points to 75, so only 10 percent of your 50 requirements could be designated mandatory. This will force your committee to prioritize. It is often helpful to have someone who is not invested in the outcome to help your team set priorities. This can be an analyst or facilitator from another unit on campus or an external consultant.

WRITE AND SEND OUT THE RFP

Even if you're not required to write an RFP, you might find it helpful to have one. Circulating the RFP draft internally can help you identify missing requirements and clarify the requirements' priority. And vendors will be forced to give you a written response stating whether they can meet your requirements. This forms the basis for a contract between you and the vendor. If you've documented your requirements, the RFP should be easy to develop.

It's often helpful to review sample RFPs from comparable institutions or from several vendors. You may see features or ways of describing requirements that you hadn't thought of. But resist the temptation to simply stick your name on someone else's RFP. The RFP needs to describe your needs, not theirs.

REVIEW RFP RESPONSES

The selection team will need to read each of the RFP responses, discarding those that fail to meet mandatory requirements and rating the rest. They need to understand each requirement in the RFP so they can rate the responses uniformly. If you use a numerical rating scheme, the team needs to apply it consistently. And they need to understand the selection process, timeline, and their responsibilities. The project leader is responsible for making sure the team is prepared for its role.

When the team reviews the RFP responses, it can assign points to each requirement, as described above. Or each item can be rated based on how well it meets your requirements, using categories like "met," "partly met," "is expected to meet in next release," or "didn't meet." These categories can have point values — e.g., if the requirement is worth 10 points, then "met" is worth 10 points, "partly met" might equal five, "next release" might equal two, and "didn't meet" gets no points. After the points are assigned, any weights are applied. Each vendor who met the mandatory requirements would be ranked based on its score, and the top vendors, perhaps three to five, would be invited to demonstrate their products.

Note that price isn't necessarily a screening factor at this point. Some RFPs instruct vendors to submit prices separately, or at a later date, so the first review can be based solely on features.

SCHEDULE PRODUCT DEMONSTRATIONS

The selection team must attend all demos. But you should also invite a wider audience and ask for feedback. The selection team might want to have the audience fill out a survey so it can get opinions on specific features. The team should also have its own rating scheme so it can score the demos consistently.

Vendors will have "canned" presentations that they are prepared to give. You might also want to request "scripted demos." In scripted demos, the selection team creates scenarios for the vendors to demonstrate. For instance, you might have vendors create some user accounts and set up their security profiles. Next, they might create profiles for some alumni, donors, and parents. Then they might show you how to enter a variety of gifts that match different scenarios (e.g., hard versus soft credit, matching gifts, joint gifts, split gifts, multi-year pledges, campaign gifts, and so on). They might "marry" some alumni and show what happens to their donor histories. Then they might divorce some of them.

Be sure to allow time for the vendor to do a canned presentation to show features you're not asking about. And allow plenty of time for questions. Demos usually last two to four hours and can easily take a full day. You might want to schedule separate demos for different groups: the computer staff will want to see how the system is configured and test the security, while the fund raisers will want to see the donor and prospect management features.

DO YOUR HOMEWORK

After the demos, you should have narrowed your choice of vendors to two or three. Now you need to verify your impressions.

Check references

Each vendor should provide you with several references. The selection team leader should contact each reference and ask the same set of questions. She might also try to contact other clients.

Visit client sites

Once you've narrowed your choice to one or at most two products, consider sending the selection team to visit comparably sized offices that have the software running. These sites have survived the conversion and can show you how the product works in real life. They can point out the hazards you're going to face and offer survival tips. The team can see how the software performs with a real user at the keyboard. And, as a side benefit, the team will meet people who are likely to be valuable resources during your conversion.

You need to view site visits and reference checks with a critical eye, however. It's important to distinguish between a poor software implementation and weak software. On the other hand, just because one college loves a product doesn't mean you will.

Make Sure you Understand the Full Price

The price of the software is just one piece of the "total cost of ownership." You're likely to need to upgrade your server, and may need to upgrade your desktop computers and office automation software. You might need consulting help to implement the software. There will be annual maintenance costs for your software and hardware. You might need to invest in additional training or enhance your technical support organization. You may initially see these as one-time capital expenses, but many will become ongoing operating costs.

Ask lots of questions

The following questions can get you started when you check references or visit other clients. If possible, send the questions you'd like to ask ahead of time.

About their implementation project:

- How long did it take for you to "go live" on the software?
- How many of your staff worked on the implementation?
- What were their skills, roles, and job titles?
- Did you dedicate any staff to the project full time? Did you have to hire temporary staff during the project?
- What did the vendor do to help you with the implementation?
- How did you organize the implementation team(s)? How many teams did you have; how many people were on each; and what were the teams' roles?
- Did you use consultants? If so, for what? How much did it cost? Was it worth it?
- Did you contract with the vendor for additional implementation support?
- Did you try to improve operations as part of your implementation? If so, how extensive was the process improvement project, and at what stage did you do it?

About the data conversion:

- Who handled your conversion programming?
- What weren't you able to convert?
- Were you happy with the vendor's support during the conversion? If not, why not?

About the software and vendor support:

- Have you required any customization of the system?
- How many software upgrades (patches, bug fixes, new releases) do you usually get each year? How much time does it usually take to install them? Do you usually have to shut the system down to install upgrades? Do upgrades tend to introduce new bugs?
- How quickly does the vendor respond to technical support questions? Are problems resolved to your satisfaction?
- Were you happy with the training provided by the vendor? If not, why not?
- Did you have to buy more training than you'd anticipated?

About the ongoing support of the system:

- How many technical staff members support the system? Where do they report? What are their titles and roles? Did they have to learn new skills in order to implement or support the system? Did you have to hire new staff to support the system? Did you have to reclassify your technical positions to retain or recruit staff?
- How many functional staff members support the system in each unit? Did you have to create any new positions or change job descriptions or classifications?

- Did you have a help desk before the implementation? Do you have one now? If you do, how is it staffed? If you don't, who answers users' questions about the system and troubleshoots problems?
- What other investments did you have to make as a result of the implementation (hardware, software, networking, training, etc.)?
- How are you handling ongoing training?
- Did you experience unusual staff turnover during the implementation or after you went live?

About the daily use of the system:

- Are fund raisers, alumni officers, and other senior administrators able to do their own research in the system?
- Can administrative staff members write their own reports? If so, what tools do they use?

About your operations generally, and what you've learned:

- What would you do differently if you had it to do over?
- What would you do the same way?

NEGOTIATE

Based on your requirements analysis, the RFP responses and demos, reference checks, and site visits, you may identify gaps in the software or areas where you'll need additional support. You may be able to negotiate with the vendor over costs such as additional training, conversion support, or software modifications. If you're buying based on the expectation that the next product release will include new features, or that the vendor will meet specific deadlines during the implementation, you might build performance incentives into the contract.

You'll want a legal opinion on the proposed contract. You might also want an opinion from a consultant who is experienced with software contracts. S/he can provide assurance that safeguards are in place and pitfalls are avoided. For example, they can help you define what it means to accept the software, or help define what type of problems might be found and the vendor response time that would be appropriate to address them.

TAKE A DEEP BREATH

The hard work — implementing the new system — is about to begin. Before you start, complete the selection process by documenting *why* you selected this system, *how* the selection process occurred, and *who* was involved. Then communicate your message broadly. If you've started your implementation planning, include as much as you know about the schedule and project team. Be sure to thank everyone who was involved in the selection process. Progress reports such as this will help you gain buy-in and support.

The Ingredients of Packaged Software Implementations

Congratulations! You've bought a new alumni/donor database. No matter how simple or complex your selection process was, you should mark this important milestone. A small celebration will help build enthusiasm and energy for the implementation — which is likely to be long and grueling.

Implementations are complex projects. You'll need strong project management, a focused team, lots of communication, and a plan. This chapter outlines the major steps in implementing new software.

Determine the Scope
Most software packages have more functionality than you will be able to implement immediately. Some may contain features that your institution is not interested in at all. To keep the implementation on track, ask yourself these questions:

- What are you trying to accomplish?
- What will success look like?
- How much time do you have?
- What's your budget?
- Are you planning to reengineer processes as you implement? If not, should the team spend any time trying to improve operations?
- When do you move from implementing the system to operating it?

Beware of "scope creep." For instance, will upgrading your department's desktop computers fall within the project's scope? What about computers for other campus departments that will be using the system? Would a network upgrade for your office be within scope? What about upgrading the network between you and the campus computer center? Scope creep can undermine your timeline, your budget, your confidence in the project, and your staff's morale.

Think About Process Improvement
Implementing a new system will invariably require changes in the way your office operates. This can be a great opportunity for process improvement. Encourage staff members to question the way they've always done things and to use the new system to its fullest.

Process improvement often takes the form of a process-reengineering project to diagram workflows; trace handoffs of paper and responsibilities; and eliminate inefficiencies, unnecessary steps, and redundancies. Large organizations usually require formal meetings and the assistance of a facilitator. But your project team can often identify simple process improvements as they determine the codes and steps necessary to run the system.

Assemble the Team
You need to identify the people who are going to get the job done. The staffing of an implementation project will vary considerably depending on the size of your institution and the complexity of your project. Minimally, you will need a project sponsor, who provides executive-level support, and a project manager. You might also need a steering committee to provide broad participation in decision-making, and multiple project teams.

Once you appoint the team, you need to make sure they understand the goals and scope of the project and their own roles and responsibilities. If your college is small, all these roles might have to be filled by one or two people.

The primary tasks to accomplish during implementation are

- determining how the system will integrate with your business processes (i.e., how will work get done using the new system?);
- preparing the technical environment;
- defining all necessary tables and values used by the system;
- setting security and access parameters for use;
- building reports and queries;
- converting data from existing systems;
- developing documentation for operating procedures;
- preparing and delivering training for users; and
- establishing maintenance procedures for the system after you go live.

Here are descriptions of the major roles of the implementation team.

Project sponsor
This person
- is often the least involved in the details of an implementation but most critical to its success.
- should be a member of upper or executive management, and in most cases should not come from the technical side of the house.
- champions the project from cradle to grave. She sends the message that this project is critical and is worth the effort and disruption. The sponsor makes sure the project manager has the resources she needs, and keeps other executives informed about the project.
- is the ultimate authority for removing roadblocks and resolving conflicts.
- is the project's cheerleader and patron. When implementation projects falter or fail, it is often due to poor stewardship by the sponsor. Very little that the project manager can do will overcome a lack of leadership in this critical role.

Steering committee
In a small office, the project leadership might consist of one or two people. But larger institutions generally require a steering committee, particularly when the project involves multiple departments. The steering committee will

- guide the project and make sure affected parties are consulted;
- make high-level decisions about the project's direction and goals;
- provide advice to the project manager and team; and
- help resolve conflicts.

The major stakeholders should be represented on the steering committee. The project sponsor is often a member of the steering committee.

Project Manager
The project manager is the day-to-day leader of the project. This person

- works with the steering committee, implementation team, software vendor, and any consultants to assemble and manage the implementation timeline, budget, and project team;
- determines and tracks key milestones and deliverables. She needs to monitor the project's schedule and keep the steering committee and sponsor informed of progress and problems;
- is often a member of the steering committee, at least in an *ex officio* capacity; at a minimum, she should give the committee frequent status reports on the project.

Ideally, the project manager will not be responsible for the hands-on work involved in the project. Nor should she have routine operational responsibilities. It is nearly impossible to stay focused on a complex project amid the distractions and daily crises of the office. Lack of time for the project (on the part of the manager, team leaders, or team members) is one of the most common causes of missed deadlines.

Team Leader
In a large university environment, you might have teams representing each department and professional school, while a small college might not need a team. If you do have teams, each requires a leader. This person

- is responsible for making sure each team member understands her role, responsibilities, and where her work fits within the overall project.
- must keep the team's part of the project on track and resolve any problems that arise.
- is responsible for communication with all other team leaders and the project manager, and for keeping the team informed of any changes in the project.
- must also support the team by keeping them motivated and helping them get resources.

Like the project manager, team leaders will be most productive if they are relieved of their regular duties. If you can't get by without them, try to reassign half of their duties.

Team Members
Team members are the most critical part of the team, yet often the least visible. Team members

- help translate day-to-day needs into the codes and business rules that drive the system.
- may also need to have duties shifted in order to have time for the project.

Think About "Buy In"
The people who will implement the system need to be committed to its success. They're going to go through a lot of hard work and inconvenience, and therefore need to know that it will be worthwhile. They need to feel that they are stakeholders in the project and that there is a decision-making structure to address their concerns. If they don't buy in to the project, they're likely to fight it, overtly or covertly, and may continue to do so long after the conversion is finished. The project will be much easier to lead if the troops are supporting you.

Create the Project Structure
Everyone involved in the project will benefit from a well-articulated decision-making process, clearly identified resources, ongoing communication, and good planning.

Decision-making process
Colleges and universities are often unaccustomed to making the kinds of quick decisions needed to keep an implementation moving. Getting decisions made can be the hardest, most time-consuming, and most expensive part of the project. The larger and more complex the institution, the more difficult it becomes. Implementations can take years if you have to get consensus from dozens (or hundreds) of departments on every code in your system. You need to devise a decision-making process that allows you to reach agreements quickly, and decide how you'll break logjams.

You need to decide what's open for debate and what's not. The project manager, implementation team, steering committee, and project sponsor must take responsibility for keeping the project moving. If your project will involve consultants, this is often a place where they can be helpful. They may be able to help you design objective criteria and a process for making and vetting decisions that are not tainted with an "insiders' agenda."

Communication
Regardless of the size of your project team, communication is critical to the success of the project. Communication sets expectations for the outcomes and conduct of the project, creates buy-in, maintains momentum, and minimizes rumors and misunderstandings. Communication should be two-way, allowing you to keep stakeholders informed about the project and to receive their input.

Good project communication is seldom *ad hoc*; it requires a plan. Your communication plan might be as simple as regular updates at staff meetings. Or it might involve a schedule of town-hall meetings, reports at other departments' staff meetings, newsletters, surveys, e-mail bulletins, Web page postings, suggestion boxes, and other ways of informing the campus and learning about their concerns.

It's a good idea to select a member of the project team to lead the communications effort. This person should prepare materials for release. The project manager or sponsor should review the materials for accuracy and consistency. The more active your project team is regarding communications, the more effectively you can manage the grapevine.

Timeline
Your software vendor and/or implementation partner or consultant, working closely with the project manager, usually outlines the project timeline for packaged system implementations. The vendor or implementation partner should know the major steps involved in the project and how long they normally take. The project manager needs to adjust for special circumstances such as the complexity of the environment; the decision-making process; budget; availability of the project team; university culture; dependence on other projects, decisions, or funding allocations; and other issues that might affect the implementation. The timeline is only a guidebook. It needs to be edited continually to match reality.

Budget
Obviously, the project budget will have to cover the cost of your new software. And you probably thought about a new or upgraded server. Other expenses that are often covered within project budgets include new forms or letterhead; licenses for applications like Word or Lotus; and report writing software, scanners, and document management systems. You may want to include funds to cover training and attendance at user group meetings. You may need additional staffing during the implementation. Depending on the software's degree of fit with your unique requirements, there may be significant costs for software customization. You may also need to budget for programming assistance for data conversion. If your organization will use consultants during the project, you'll need to add another budget line for them. And don't forget to build the annual maintenance costs of your new software and hardware, as well as ongoing training, into your annual operating budget.

Contingency plans
Of course, nothing will go wrong on your project. But what if it does? What if your new system is so popular that you need more user licenses? What if the server you bought isn't fast enough to support all those new users? What if the network can't handle the traffic? Or your project manager finds another job? Some of these contingencies can be anticipated; for instance, you might create incentives for key staff to stay with you at least until the project ends. But you also need to have contingency reserves and creativity on tap in case something unforeseen occurs.

Prepare the technical environment
Minimally, you need to install and configure your software. Depending on the current state of your technology platform, you might also need to install network cabling, networking equipment and security, servers, desktop computers, e-mail software, applications software, scanners, printers, or Web servers. Each will take time to install, configure, and test. Many institutions begin the implementation process with a technical

assessment so that they can include time (and funding) for any necessary purchases, installations, or upgrades in the project plan.

Configure the Software
This sounds like a purely technical step, but don't be deceived. This is where your decision-making process comes fully into play. The project team needs to decide everything from how you'll code addresses to what the rules are for managing donor stewardship. If your new software is complex, the team may have to make thousands of decisions. If your environment is complex, you might have hundreds of departments whose needs must be addressed. The project team needs to communicate with stakeholders, make the decisions that fall within their authority, request decisions on questions that are outside their area, and get disagreements resolved expeditiously.

Develop Interfaces
If you're implementing stand-alone software, you'll probably need to link it to other campus systems and to external systems (e.g., a telemarketing system). But you might need interfaces even if your system is part of a campuswide software package. Not every module will go live at once; you might need temporary interfaces (e.g., to the general ledger, admissions, or student systems).

Develop Reports
Your new system will come with "canned" reports. In the best of all worlds, these reports will give you all the information you need in just the way you want to see it. But don't count on it. You need to look at your current reports and decide which ones you really use. Then make sure you can get the same information from the new system — either in hard copy or through on-line inquiries. Many institutions find that the information available through inquiry screens makes many of their former reports obsolete.

You're also likely to need a way for end-users to run their own reports. This often requires an additional report writer or an interface with commercial spreadsheets or databases. And users will need training on both the reporting software and the techniques for creating *ad hoc* reports that balance with the canned reports.

Train the Staff
Even the best development software won't run itself. You need to invest in the staff members who will use the system. And you're likely to spend more on training than you'd planned.

Obviously you need to train the team when you begin the project. But six months or three years later, when you finally go live with the new system, they might need a refresher course. You're also likely to lose and/or add staff, and the new staff will need training. And upgrades to the system might be extensive enough to require still more training. A "train the trainer" approach can help you bring training in-house and contain costs. But not everyone is an effective trainer, so you need to make sure you have someone with the skills and patience to do a good job. And even with a train the trainer approach, your technical staff and “power users” are likely to require more in-depth training than you can deliver in house.

Don’t forget the population of casual users who may need training on inquiry and data searches. They may not have direct responsibility for using the system in your department, but they will benefit greatly from understanding how to use the information the system provides.

Develop Documentation
One frequently overlooked aspect of the training phase is preparation of a documented set of administrative procedures. It’s important, as the business processes are being applied to the system (and vice-versa), that you capture the flow of work with clear, concise procedures documentation. This will become the basis of your training program and help ensure that information about how the system works and how the development enterprise operates is available for reference. A good set of procedure documentation is your insurance against loss of institutional memory, which can occur with frightening speed if you lose key individuals in your department.

Convert your data
Data conversion has two major steps: mapping and programming.

Data mapping
In this step you are trying to translate all the fields in your old system into corresponding fields in the new system. This is a time-consuming process that requires communication among several parties: a staff member who knows your data, technical staff members who know the new system, technical staff members who know the old system, and the programmer(s) who will do the translating.

Identify someone from your staff who knows your data and assign her to this project. The people who handle data entry or report generation are probably your best resources. If you don’t have anyone who knows your data, find someone who is familiar with your old system, preferably from a functional perspective. This might be someone from another college or a consultant recommended (or provided) by the vendor of your old system. They won’t know the idiosyncrasies of your data, however, so they will need to ask lots of questions.

If your only choice is to use technical staff members, you must work closely with them. They probably won’t know your data, are unlikely to know a LYBUNT from a non-donor, and might not know what questions to ask.

This step will be more complicated if you have used fields in your current systems in varying ways (e.g., spouse name was used to store children’s names if there was no spouse). To the extent that you are aware of such practices, you should arm your conversion team with everything you know. This is no time to be coy about the condition of your data.

Programming
In this step your old data is massaged to match the format of the new system. The complexity of this step varies tremendously. In the best case, the data comes out of your old system just the way the new system expects it to look. In the worst case, you will need a lot of technical help.

The problems that you might face in this step include

- The new system can store all of the data that was in your old system (this is rarely the case, but let's fantasize). But some fields are too short. What do you do with the last five characters of long addresses?
- Your data has been forced into fields in irregular ways over time.
- You don't have a (good enough) programmer on staff and can't afford (or find) anyone else.
- The programmer hasn't done this type of work and doesn't anticipate the hazards.
- The documentation for the new (or old) system is so bad that no outsider can figure out how the data is organized.
- The conversion requires the use of special tools that only the software vendor can use.
- The programmer doesn't ask enough questions.

A do-it-yourself approach at this step can get you into trouble. And accepting a cut-rate bid from a hired gun will frequently backfire. If you bought a simple system, any programmer (and perhaps a talented amateur) might be able to program the conversion. But if your new system is (or your old system was) complicated, the programming is probably equally complex. Get multiple bids, and be prepared to invest in this critical step.

In many cases the vendor who wrote your new system has a huge advantage in handling the conversion programming. The vendor knows how the new system stores and organizes data. And some systems are so complicated and/or so poorly documented that only an insider can figure them out. But the vendor is often the most expensive option and isn't always the best. Some vendors routinely fail to meet their conversion schedules. Others vendors simply don't do conversions.

If you didn't ask about conversions when you checked references on your software vendor, locate its user group and start asking. If someone wasn't happy, investigate. Was it the vendor's fault? If so, was it symptomatic or an isolated problem?

If you decide to look elsewhere for programming help, try to find a consultant with experience handling conversions to your new system. Some software vendors are happy to recommend consultants. If they're not, try asking through their user group. You may also want to look into firms that provide "data scrubbing" that can, at the least, remove duplicate records and prepare your current data for importing to the new system.

Since development operations are so highly data-dependent, it is critical that you use the implementation conversion to "get it right." After cleaning the data and preparing the conversion plan, be certain that your standards and formats are documented and added to your operating procedures. This will help ensure that your new system's data stays clean and usable.

Test everything
Now you're ready to make sure the previous steps were done properly. Do the dollars add up? Are dead donors still coded "deceased"? Are alumni couples still linked? Do the new reports balance with the old? Can you read them? Are the screens laid out logically? Can users move around the system in ways that makes sense? Do reports come out on the right printer? Is the system security working?

Remember, of course, that any garbage from the old system is still garbage in the new system. Data cleanup is another project altogether. If you didn't attend to it during the conversion, you should plan to devote significant resources to it during or immediately after your conversion.

Who should do the testing? Everyone. The programmers should have been testing all along. Now the people who did the data mapping should look at sample screens and reports. So should fund raisers, alumni officers, researchers, data entry staff, and receptionists — anyone who knows your data or constituents. Look up your own record and anyone else's whose information you're familiar with. Run the same reports on the old and new systems and compare the results. If something looks wrong, ask lots of questions, get problems fixed, and test again.

Consider running parallel systems for some period of time. This means entering the same data on your old and new systems and making sure everything balances on both. This can be a good training exercise before cutting over to the new system. The duration of parallel testing can range from one gift batch to a full year, but few development offices run parallel for more than a month. You probably won't want to run parallel during a busy time of year, but you should try to test a range of scenarios.

Flip the Switch
Be prepared for minor upsets. Most people take a great deal of pride in their work, and having new systems to work with can disrupt their sense of accomplishment. Under pressure to perform, it's easy to feel overwhelmed. The transition will go more smoothly if everyone allows a little more time in the first week or so.

Celebrate!
Your staff has been through a long, difficult project. You need to acknowledge their hard work, and let them know it was worthwhile. Plan to end the first week with a party to celebrate the cut over and publicly thank your team members. Invite your sponsor and other officers to join you and be sure that they also take time to thank the team.

Take Stock
Going live on a new system doesn't mean your implementation is over. The project team will have a long list of details that aren't finished. And the technical staff members are about to be deluged by requests for new reports, screens, codes, alumni regions, honor

rolls, labels, tapes, pledge rosters, receipt programs, and so on. They're going to need a process for prioritizing requests, communicating with users, fighting fires, working on long-term projects, and balancing ongoing implementation with their daily workload.

As these mechanisms are worked out, it's a great time to make the transition from the project manager to the systems support team. Schedule regular meetings to make assignments, report progress, and continue the integration of your new system into the daily operations of your office.