How to Select Business Software
George J. Miller, CFPIM, President
***PROACTION* Management Consultants**
Oak Park, California, 91377 USA

I. INTRODUCTION

How to select a major business software package?

Software selection is very complex and intimidating to the folks stuck with the unenviable task of making "the right choice" for their companies. Business lore is full of disaster stories of wrong decisions or indecisions resulting in paralysis, or ineffective half-measures that help destroy a company's competitive strength. This paper covers the major issues and presents recommended approaches to accomplish the task and help ensure success.

Selection teams have great difficulty grappling with or even identifying some issues related to software selection. It is frequently and erroneously identified as a technical task. Although there are in fact more technical issues to grapple with than ever, ***software selection is still more of a business and political task than a technical one.***

The selection event itself is but a part of the entire job. The spectacle of numerous companies perennially "shopping" for software, without proper objectives, goals, support, needs definition or coherent plans alternately amuses and shocks one. These items must be addressed first to set the stage for successful selection and subsequent implementation.

This paper deals with selection of “packaged” software. It is assumed that most companies will have integration and some customization requirements. Largely custom projects have additional dimensions of complexity beyond what is described here.

Time is a major factor. Every month that large projects go on without completion means major lost benefits and higher expense. Most selections could be cut drastically in duration.

All of these have always been problems. In addition, there are some newer things needing consideration . . .

Change #1: In recent years the selection job has become even more difficult, with increasingly broader applications scope involving more

departments/processes and functions with a stake (and vote) in selection, and more complex, varied operating environments. It is no longer a case of selecting “ERP” software for Manufacturing, a quality statistics “package” for QA, "financials" for Accounting or an order entry system for Sales. Instead, a more strategic, integrated, business-oriented, enterprise-wide and “system architecture-based” decision needs to be made for best future prospects of the organization.

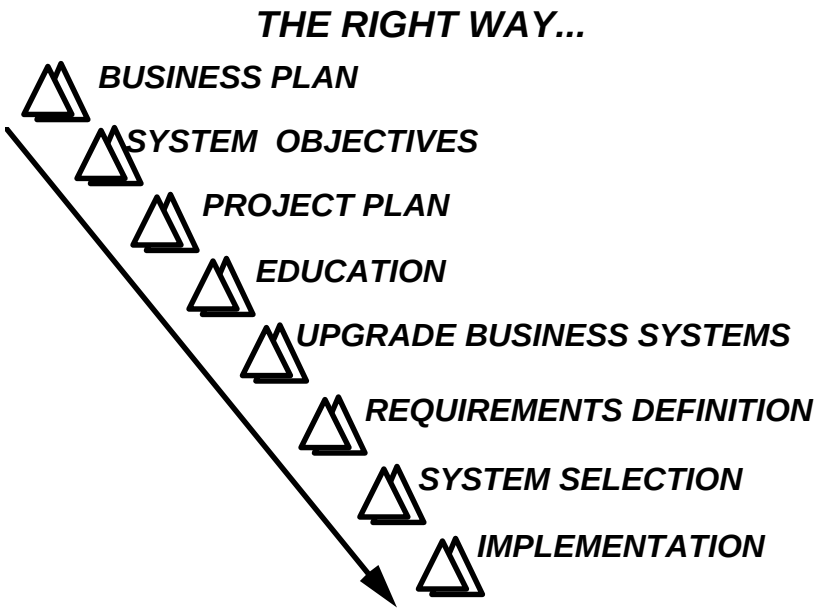
Change #2: With the rise of Enterprise CRM (Customer Relationship management), E-Business, SCM (Supply Chain Management), ERP (Enterprise Resource Planning), JIT (Just-In-Time), Lean Manufacturing, Rapid Time to Market, TQM (Total Quality Management), CIM (Computer Integrated Manufacturing), TEI (Total Employee Involvement) and other sweeping changes in the way business enterprises are run, there are now more choices of management philosophies "imbedded" in software and new policies/procedures to consider as well.

Change #3: Selection criteria need also to be "deeper," due to greater reliance on software to handle more work, mission critical systems, 24/7/99.9999% availability, and more extensive "hard" integration with other systems and equipment, through electronic commerce, Internet/Extranet/Intranet, DNC, graphics/text integration, client workstations, servers, networks, data collection, CAD/CAM, etc.

Change #4: Finally, criteria need to be more sophisticated in other ways, such as considering "friendliness" and technical issues, like "windows", browsers, "navigation", on-line help and procedures, parameters, object oriented design, new platforms, interconnectivity, recovery, security, audio-visual features, accessibility and even imbedded artificial intelligence. Business modeling and workflow features are revolutionizing business systems. The overpowering influence and rapid rate of change related to the Internet MUST be considered in almost ALL systems efforts.

Therefore, multi-disciplinary selection teams need to adopt evaluation techniques for newer software and needs, which confound traditional selection approaches. Requirements definitions need to be more comprehensive in scope of applications, management approach and technical considerations, but less specific in terms of how problems should be solved, to allow for different, better approaches.

Along with these newer requirements, age-old tasks continue to exist . . .



Major Task Checklist:

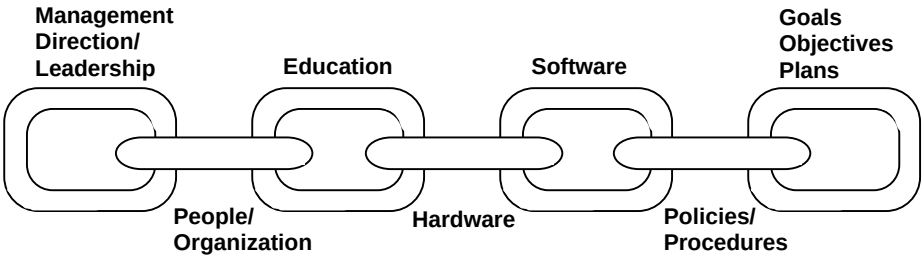
- Form top management steering committee, develop mission statement
- Set up project team
- Review business plan
- Write project charter, objectives, operating rules
- Provide general education
- Define requirements
- Justify project
- Get support: money, manpower, direction, consensus
- Start the project-- don't "wait for the software"
- Plan project (high level)
- Engage partners- System integrators, consultants
- Provide applications education
- "Clean-up" existing systems- policies, procedures, data--
- Develop supplier selection criteria (not the same thing as systems requirements)
- Survey/screen candidates through research reports, consultants, business partners, RFI and informally
- Publish System Requirement Definition
- Publish/distribute Request For Proposal
- Receive, evaluate Proposal
- Test/evaluate
  - Demos, system testing, proof of concept
  - Study documentation, proposal
  - Visit users
  - Check references
  - Investigate supplier background
- Construct technical systems architecture
- Budget

- Negotiate
- Contract
- Conduct acceptance tests (depending on additional required development)
- Make implementation plan more software-specific
- Continue with implementation.

Now more on how to play the selection game.....

II. WE HAVE MET THE ENEMY AND HE IS US.

Before you worry about what the software suppliers offer - or don't offer - get your own house in order. **Most projects that go awry do so because of internal company errors, not software supplier problems.** The software by itself is only part of the total solution to your business control problems. Some experts even say that software is only a "C" item. We don't agree. Software is becoming one of several mission-critical links in a chain, as shown in figure 3 . . .



"A Chain Is Only As Strong As Its Weakest Link"

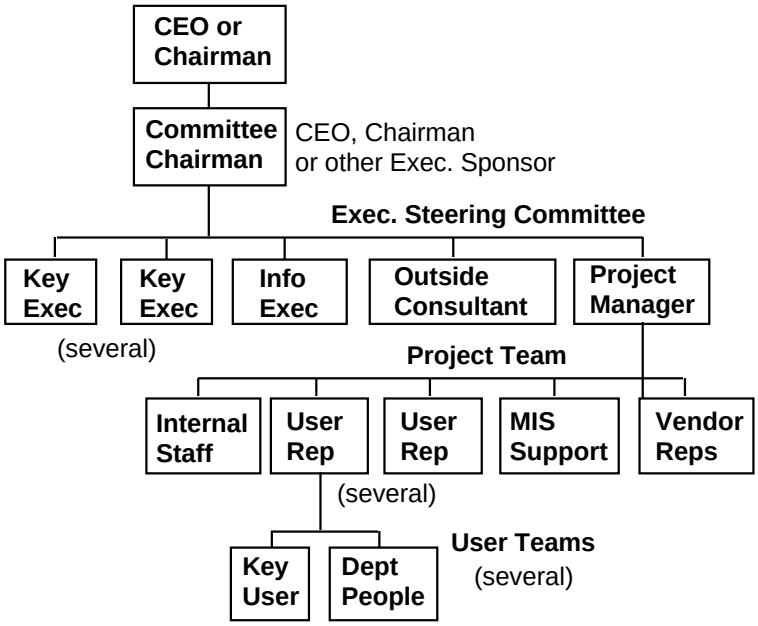
Figure 3

A. Getting Organized

1. **Don't wait until software selection time to do the things you should have done long ago.** Review the company business plan (you DO have a business plan, right?). Define project-specific goals and objectives, initiate record accuracy programs, improve things such as: customer service policies, pricing systems, bill-of-materials, policies/procedures, organizational improvements and that new chart of accounts you've been meaning to implement. Installing new software is no substitute for such more important and cost-effective things that should be done first in any case.

2. **Don't try to automate the mess you already have.** If your current system is working so great, why are you out looking for another one? Guard against the tendency for people to think in terms of what they already know. Chances are there are improvement opportunities in systems approaches, data collection methods, editing, visibility of information, improved processing and integration of information. Don't let me second-guess you though -- do your own analysis first. Formally defining business systems goals, objectives and proposed workings, then using these as a standard for selection and *implementation,* is the best way to avoid this syndrome.

3. **Don't neglect to organize for success.** Proper structuring of the effort is critical to realize your objectives. Start with an executive level team, committee, action group or whatever you want to call it. These people become responsible for developing objectives, approach, building a mandate, initiating action and guiding it along the desired path. A strong, full-time project *manager* is needed, preferably one who knows the company and its systems inside and out. This person should ideally be a user of the system, from one of the major departments involved in the implementation. The PM should probably report to the head of a key user department, or higher, or better yet, be one of those people. Sometimes, qualified IT people are in charge, or even outsiders, if there are no suitable internal resources. For an enterprise-level system, executive support is practically mandatory. The only time it usually succeeds in the absence of this is through a tightly cohesive team of management peers and/or an unusually gifted and respected lower level person- not a typical situation.



Business Systems Project Organization

Team members should be drawn from the major areas using the system, as well as “upstream” and “downstream” of the processes in question, and be of sufficiently high caliber to adequately represent those departments and command the respect of company personnel. Also use outsiders to leaven the team with objectivity and additional skills that may not be available in-house. **Use the best people you can find, not those that can be spared.** Better to have a few competent and effective people than "a cast of thousands." They need to be available when needed if the system is to be selected and implemented effectively and on-time. Decisions made here may impact the company for years.

Setting this up right from the start will help place accountability and "ownership" for the system where it belongs. **By putting the project under the people who will ultimately be using it and getting those people to sign up to the objectives and make the decisions, you will effectively eliminate most objections and "I-told-you-so's" that haunt some projects.** The team members in turn will "sell" the project to other company personnel.

Companies undertaking large, complex projects often need system integrators and/or consultants to help ensure success.

What is management "support?" How do you get enough of it? I usually smile to myself when I see some expert preaching about the need for management support for a project. There is no such thing as unqualified support. You will be lucky if you get the money to finance the project, the time to do it, and a few encouraging words and help when you are in a pinch. Management has plenty of other things to keep them busy and your project may not be foremost amongst them. Consider yourself twice-blessed if one or two of the company honchos actively aid and abet the project on an ongoing basis. A businesslike approach and regular results as advertised tend to build "support" along the way. **The best way I know of to get management "support" is to get promoted and provide it yourself. To augment management support, you need to begin building a system of informal alliances among peers and educate them to win support and get things done. Another good way is to provide results as advertised, on a regular basis-- success breeds support.**

It is more important to get key players to agree on a specific course of action rather than make the "perfect" software decision. Reasonable consensus and harmony on even a mediocre decision is better than dissent and disharmony over an excellent one. When people agree on something, they are more likely to give it their best efforts and make it work.

4. **Don't fail to determine the true costs/benefits.** No matter what people say about the intuitive "need" for a system, don't believe it. The "need" will last until the next "end-of-the-quarter" shipment hysteria or until another worthy project competing for the same resources comes along, and there are plenty of these. Perform a cost/benefit analysis and apply it well. This exercise has several benefits:

- Recommit resources
- Determine and prioritize implementation steps for best payback and political support
- Quantify decision data and help rationalize "intangible" justifications.

5. **Don't fail to create a realistic project implementation plan. Unless you determine major milestones and activities, with deliverables and due dates, the project will drag on and there won't be criteria to even measure the slippage.**

Once an overall timetable is completed, a more detailed project plan should be put together. **Do it <u>now</u>**. Most of it can be done *before* knowing who the software supplier is. It will force you to think more deeply about work to be accomplished. Many steps need to be taken regardless of which software is selected. You can always change it later.

Think of the project plan as a living, breathing document, which will change constantly, based on changes in objectives, circumstances, available resources and your perception of the tasks to be accomplished. You should maintain a very detailed plan for the next few months of activities and less detail for activities farther out in the future. As the time to perform the work comes closer and as tasks become known in more detail, you can then fill out the plan to include all known work to be performed. It is important that you break tasks down into sufficient detail to assign to a specific person and have specific, measurable, deliverables, due on specific dates. **Be sure to include start-up, requirements definition and selection tasks in your plan.**

B. Get educated

Early on in your project, ensure that key players - executive management, project team, key users and key MIS people - get outside education on "generic" integrated business systems and relevant specific application areas such as CRM, APS, MRPII, configuration management, business planning, tool management, XML or whatever it is you're trying to introduce/upgrade. Look for industry-specific or at least process-specific approaches, such as Aerospace/ Defense, Automotive, Wholesale Distribution, Chemicals, Job Shop, Repetitive, etc., since they're more relevant. In addition, general education on project management, change management and TQM will let you undertake your project with more effectiveness. Also use research services to educate your people.

C. Find out what's out there

Without getting into detailed analysis of hundreds of suppliers offering software and related services, ask around, talk to industry experts, get literature from the suppliers, research and study what's happening with business software in your chosen applications target area. Don’t clip every ad and respond to every telemarketer and brochure you encounter-- there aren’t enough hours in the year! In this manner, you will know early on what's available and possible, which will help you to properly set expectations and then . . .

D. Prepare a System Requirement Definition (SRD)

This document should become the *manifesto* for your system implementation. Not only will it become a key part of your RFP (Request for Proposal), but it will serve as the internal definition of what is to be accomplished and what approach might be taken. I won't get into detail on how to do one here, but suffice to say

that this step is key to rational development of a systems implementation. A good SRD probably includes:

- Objectives and Issues
- Company Description
- Project Charter
- Timetable
- Rough-Cut Cost/Benefit Analysis
- Functional Organization Charts
- Functional Flow and Descriptions
- Detail System Requirements (NOT specifications) Description
- Technical Requirements (Hardware, Software, Network)
- Statistics (# parts, BOM's, transactions, users, etc.)
- Glossary of Terms

When you have enough information available, it would be a good idea to send out an RFI (Request For Information) to help screen suppliers, to get them down to a more manageable number.

E. Beware the technical "sirens"

Avoid a technical/hardware-driven decision. It is not always necessary to have a 128 bit processor with multi-gigaflop speeds, operating in a totally fault-tolerant mode with distributed user-transparent data bases and other like-minded technical wizardry. **The bottom line is:**

Does it help deliver your business requirements or doesn't it?

You probably already know that you don't run out and buy hardware first then shop for software that runs on it, but I would never guess that from what I've seen in industry. The decision should be based on your business needs, on cost, time and people resources and **last,** on purely technical considerations. However, overall long-range hardware architecture is a major strategic consideration also, so don't paint yourself into a corner by picking hardware that only meets your immediate needs. Also, strongly consider an integrated suite of applications that works harmoniously to meet your business requirements. We are well aware that the costs and delays of custom integration may well exceed the benefits of assembling “best-of-breed” solutions from pieces.

F. Use Consultants Before They Use You

1. **Don't let a consultant or system integrator select your software for you.** You have to live with it, not him/her. You know your own company much better than any outsider. Most importantly, if you let an outsider make the decision for you, you will not instill the ownership and accountability needed. **Use consultants to identify viable alternatives, educate your people, supply**

evaluation criteria, review the process and provide extra "horsepower." Then take over and take “ownership’ for your own decision.

2. **Why a consultant?** There are several good reasons for bringing an outside consultant in to help with a project of this sort:

- Help improve objectivity
- Temporarily supplement manpower
- Specialization - a consultant may be experienced in these matters, whereas your people do this infrequently, if at all

3. **The "resident" consultant-** It is all too easy for a consultant to get too comfortable and begin spending too much time on your account to the point where it is no longer in anyone's best interest. This is not a good use of the consultant's time and it quickly runs up your expenditures. Even honest and principled consultants sometimes fall into this trap and it is up to you to help keep them out of it. How to do this . . .

4. **Use objective-setting and review** to your and consultant's mutual advantage. **Both consultant and client alike benefit when there is a clear statement of objectives, and a schedule itemizing specific deliverables, who is responsible and when tasks are due.** All of this should be mutually agreed upon. Periodic meetings should be held to review the status of the consultant's work and make changes when necessary. Compare to the original agreement.

5. **Make sure your consultant is spending time on your real problems and that he/she properly leverages work done in the past with others to your advantage.** A good consultant or system integrator may be able to do a lot of the steps we have discussed and more quickly and efficiently, at that. If, for example, a consultant specializes in your area, he may already have researched the software suppliers and may already be up to date on industry issues, approaches and resources.

G. Selection of Partners

There are several ways to proceed, if your company doesn’t have the skills and resources to do a project:

1. Use corporate resources to reinforce a local effort, if the skills are adequate and they will “fit.”
2. Use a consultant/consulting firm to help educate your staff, support the selection process and provide advice.
3. If the stakes are high, consider an objective, independent consultant to act as an auditor and project “conscience.”
4. Engage software selection expertise, such as SoftSelect, ChooseSmart, FasTrack, or others.
5. Utilize research materials, such as Gartner group, Forrester, Giga or others.

6. Use a system integrator to design the approach, selection process, system design and support implementation. Some consulting firms may be able to do this as well. There are a number of large and small companies suitable for the task. Typically, an integrator will take on a "turnkey" project, doing all or part of the requirements development, conceptual design, partner/vendor selection/recommendations, detail design, coding, testing, integration, network architecture, conversion, training and implementation support. Get a clear scope, statement of work, deliverables, costs, responsibilities, performance guarantees, recourse, terms and conditions, termination clauses defined.
7. Be very careful to allow for biases. People tend to recommend what they know and what they sell. Many firms have commission and finder's fee arrangements, which are unethical if not fully disclosed up front. Get disclosure statements and ethics policies in writing, early on. Check things out thoroughly.

III. HOW TO CHECK SUPPLIERS OUT - WHAT'S REALLY IMPORTANT

A. Kick the tires first

Before you get involved in the minutiae of specific functions and features of the software, check the overall qualifications of the supplier to meet your needs.

Screen first. Narrow down the field from hundreds to, say, 10-20 who appear to address your type of company and problems. Send them an RFI (Request For Information, NOT an RFP, which is discussed below) screening questionnaire requiring minimal work on their and your part to see if there is a possible fit. Some of the answers to questions suggested below may require verification later, once you get more serious by narrowing the list down to 3-5 real candidates.

1. Is the supplier committed to the type software you need? Is it a main line of business? What about addressing your specific industry or process niche?

2. Is the supplier well-managed, established and financially sound?

3. Does the product effectively address most of the targeted business problems? Is it integrated? If not, can it be effectively and economically enhanced to do so or used in conjunction with other packages that will? Specialized solutions for particular industries such as repetitive manufacturing or government contracting are evolving that will reduce or maybe someday eliminate the need to do this.

4. Can the supplier provide *you* with effective support?

- Planning, installation
- Training
- Technical

5. Is the system sufficiently flexible for expansion, interfacing/integration and modification?

6. Unless the software already does everything you will ever need, is an ambitious upgrading planned and in progress? Is there a good past record of the same? What about technology modernization to improve ease of use, speed, maintenance and interconnectivity to other systems? Do REAL partner relationships exist, to deliver things, to fill the gap that this supplier cannot?

7. Is there a satisfied user base? Although it is not necessary to have a huge installed base to prove that a system works, there should be some firms successfully using the *standard* version of the system that you are looking at. You may find that the products you like may be newer and not yet have a large number of users. This indicates that you should proceed with caution, but does not rule out a supplier.

8. Does the system employ viable hardware and systems software? Make sure it runs on something that will be around five years from now and uses supportable systems software: languages, data base management systems, operating systems, etc. Consider products that utilize so-called "open architecture" that run in a multitude of hardware/software environments. Or at least, make sure your suppliers have a good solid upgrade/"migration" path already in work, not just talk. Platform independence would be nice.

9. Does the system have excellent, easy to use documentation? (User, systems, operations, programming)

10. Is ownership or perpetual license provided to you? Is guaranteed maintenance available in the future? Do you get source code or a guarantee that source code will be available if the supplier no longer supports the product? If you ever intend to modify the product, or wish to keep it compatible with newer systems software, then you need source code or a supplier who will make the changes for you.

11. Is there an effective user group? This is usually a sign of a viable package that is actually being used by customers. Beware of "window dressing" groups set up by software suppliers for marketing purposes. Beware of groups that are merely "social clubs" for low-level user personnel. *Next step . . .*

B. The Request For Proposal (RFP)

This is needed to define requirements for, form the basis for evaluation of and structure the relationship with suppliers.

Don't send out numerous RFP's to every supplier in the world. It isn't fair to expect suppliers to put in the great amount of time required to answer one of those monsters with little hope of making a sale It makes poor use of your own

time to undertake a detailed analysis of so many suppliers when you will ultimately be using only one, in many cases.

Those in the software business marvel at prospects who spend several times the amount of the software cost just to put together an RFP. However, a well-done and reasonably priced service is well worth a good sum of money, when you consider the true lifecycle cost of a major system, particularly if it's done wrong.

Don't let the RFP become a "wish list" of all the functions and features that a software package might ever have, but rather make it focus on your specific requirements. A good RFP is basically a "sanitized" and expanded version of the System Requirements Definition (SRD) described in section II-D. Some additional sections are needed:

- Response Instructions and Schedule
- Supplier Qualification Standards (where appropriate to disclose)
- Terms and Conditions (yours, not just *theirs*)
- Pricing Formats

If designed properly, it becomes a turnaround document for the supplier to respond directly upon, requiring only supplemental attachments, where appropriate.

Structure the RFP response rules so that the supplier can come back with creative solutions. Don't force a simple "yes" or "no" answer for everything. The honest suppliers will end up disqualifying themselves, and the dishonest suppliers will answer "yes" to everything. Make provisions for allowing suppliers to answer questions with alternate solutions, qualified answers and discussions of future enhancements (which you will take with a grain of salt) as well as those currently available. **Be flexible**- probably no package will meet all of your requirements. Be prepared to give up something to get something you need even more.

Make fraudulent answers grounds for disqualification. Even better, make the RFP response and all related documents part of any negotiated contract.

C. Getting Serious

Once you get down to a couple of real finalists, you're ready to really dig in . . .

1. **Check the place out.** Meet the management. Not just the folks at the local sales office, but the headquarters personnel. Go there. Find out the background and orientation of the principals/executives of the firm. Become familiar with some of their policies, procedures, standards, discipline and apparent adherence thereto. Meet the actual people who will support you, too.

Find out where the money is coming from to run the place. Demand to see financial statements when you are farther along in the qualifying process. Talk to employees, suppliers and customers. Have members of your own upper management check them out as well, through their own sources. Not only will this strategy uncover more information, but it will serve to bring management closer to the project and may help uncover their pet or disliked supplier prejudices, if any.

2. A crash course on the economics of the software business: Marked similarities - and differences - between software and manufacturing firms exist. Know this and profit from it. **Software company executives have "end-of-the-month/quarter/year" shipment pressures too!** That's the best time to make a deal, but be prepared to "close" quickly.

It isn't true that the higher the ratio of development/support expenditures to sales expenditures, the better, a prejudice we sometimes see in selection criteria. Successful software firms usually spend more on marketing/sales than on development. They've got to constantly feed the monster with revenues to expand and improve their product.

To control development costs, a supplier must judiciously lay out product plans to become strong in areas of chosen **focus**. In software, like other businesses, a supplier cannot be all things to all people. **Even the largest software suppliers cannot create systems that deal with all business environments on their own.** It is common for them to work with partners to provide portions of the overall solution. Just because a vendor acquires other companies with complementary products doesn't mean that they are able to assimilate, integrate and support them adequately.

Suppliers who have been in business a while earn substantial revenues on ongoing maintenance contracts and the sale of new products to existing customers. Newer suppliers must rely upon selling a number of new customers to stay in business.

Make sure the supplier is:

- Keeping current on technical and applications features
- Staying price and service-competitive
- Getting and keeping referenceable accounts by offering quality products and services
- Minimizing custom software development commitments for specific customers
- Achieving sales and profit projections to satisfy commitments to stockholders or investors
- Containing installation and support costs.

Understand their problems and you will understand what is possible for you and what isn't. Structure the relationship with them based on your knowledge of their situation.

3. **The quickest way to check on a supplier is to talk to customers**. Not just the hand-picked, cultivated references given by the supplier, but your *own* independent contacts within customer organizations, who are more likely to be frank with you. Find out how well it works and what kind of glitches exist.

Do your own checking too. Check functionality and ease of use.
Have your people look at the systems documentation, programming documentation and the program code.

D. Beware of "features poker"

1. **Most software packages are not "basically the same"** with minor incremental differences, in spite of what you might have heard or read. There are *major* differences in the quality, scope and approach of different suppliers.

2. Beware of marketing hype versus the real product. This is **especially** true of some of the newer "vertical market" solutions. A number of suppliers are jumping upon the “B2B,” “B2C,” Just-In-Time, CIM, repetitive manufacturing, automotive and government contract bandwagons right now, claiming to have "solutions" in areas where they do not. Beware of very new products claiming to be extremely comprehensive and well-proven. They aren't.

3. **The "skeleton" module:** Some suppliers offer a truly awesome number of different modules, functions and features on paper. This is sometimes accomplished by providing only bare minimal functionality in a given area and then trumpeting that feature as a complete "module" of the software. Naive project selection teams are sometimes taken in by this technique and only find out belatedly that the fifty module system they think they bought turns out to be a paper tiger.

4. **The "functions and features" trap.** The bewildering array of features offered by some suppliers tends to bedazzle the novice project team. Most companies would never use or need some of the features offered. **The best way to stay on track is to have a formal definition of your true requirements in the System Requirement Definition, previously described.** Project teams frequently get tied-up in the mechanics of trying to evaluate large numbers of features while ignoring some of the important qualification criteria outlined earlier in this paper. Avoid blind use of numerical "point" systems focused on feature evaluation. Flexibility, usability and support are just important as functionality.

5. It's in "the next release-" Also see "the check's in the mail" and "this won't hurt a bit." When you hear this - check your wallet. It may be perfectly true, but consider this: the best intentions can be diverted by other priorities,

manpower or skills may be inadequate, software designs may have to be re-done, and testing and debugging can take far longer than originally anticipated. Unscrupulous salespeople may indicate that the desired feature is just around the corner when it is in fact not. Even if the supplier puts it in writing, satisfy yourself that they in fact have the wherewithal to actually deliver it. Even better- check what's available *now* and how they've performed to promises in the past.

6. Wolf in Sheep's clothing— The vendor presents it as their standard product, but it's actually a highly customized version that the customer spent extra megabucks on to do it his way. You will never see this product run in your shop.

7. **Demo sleights-of-hand**- Skilled and motivated salespeople can make software appear to do nearly anything. They know how to stress the good points of systems and how to avoid or gloss over weaknesses. This you can expect from the honest salespeople. Less scrupulous ones may actually "rig" demos to make it seem that systems have capabilities which they do not. Make sure that the demonstration you see is the *standard* software, not a special or test version which is not made available to you. **The only way to really be safe from these horrors is by reference-checking and conducting your own tests, under your own control, on your own terms, in your own time.** This is covered in more detail in Section IV.

E. The "friendliness" issue

1. "User Friendliness"

- **On-line versus batch systems-** There is no substitute for on-line, interactive data entry and update capability available at the point of origination of the data. Using the system in a conversational mode brings the "user" much closer and permits development of better understanding of and "relationship" with the system. Some functions, of course, are better with replicated data or periodic batch updates, where it makes good business sense.

- **User interface -** Conventions such as "intuitive" operation, color graphics, icons, pull-down menus, windows, scrollable screens, drill down, audio/visual features, standardized protocols and hypertext/hyperlinks make systems far easier to learn and utilize. Windows generation people are insisting on these amenities and are now getting them. Cross platform industry standard browsers are now an increasingly accepted norm.

- **Workflow-** The system should be readily adaptable to best practice processes (not necessarily the ones you have now) and encompass manual and automated flows across multiple applications and organizations. This may require restructuring your processes, organization, even facilities, eventually.

- **System integration** - Combining of manufacturing, financial, engineering and sales functions into a single unified system. All programs in the system update a single logical database (although there may be multiple physical databases), which in turn are used by the entire company. All related tables are updated simultaneously, where applicable, and in turn can be made available to supply information when needed. There are cases when you don't want it to be a single physical system. For example, you might separate your web-based order entry system from the "back-end" system," to maximize uptime and security.

- **On-line help/documentation** - The best software provides easy-to-use help functions and documentation manuals callable right from the screen. Better suppliers provide procedures as well and allow you to customize them. It is becoming common to see built in implementation tools to enable procedure "mapping," customization of implementation options and documentation.

- **Not too complex -** In spite of the increasing complexity of manufacturing systems, operations shouldn't be too hard for the people who are *using* them. Even though there are many interrelated functions within a system, the operation of any one transaction should be a simple, efficient and trouble-free experience and it should be easy to find what you need without a guru at your side.

- **Documentation** - Clear, complete, up-to-date documentation - user, operations, systems and programming. On-line access and maintenance preferred.

- **Users firmly in control** - Nearly all system functions should be operated and controlled by the appropriate users of the system. They should not have to rely upon data processing nor the supplier for any but the most arcane, complex functions.

- **Audit trails, back-up and recovery** - System must be capable of providing a record of the source data that caused the database to exist in its current state, and be able to back-up and recover the data base in the event of a malfunction. Archiving and restoration tools are needed. A disaster recovery scheme should be waiting in the wings.

- **Fast -** Response time and performance need to be rapid enough to make economical use of user's time and avoid frustration. Get specific, *written* contractual promises.

- **Access to data in user-determined format** - the ability to create and update screens, inquiries and reports quickly, easily and economically is vital and now expected in today's business systems world.

2. "Technical friendliness"

- **Data base management systems** (DBMS)-
  Standardized methodologies for structuring relationships of and maintaining data in the system are nearly mandatory in today's complex manufacturing systems. Although it is not mandatory to use a commercially available system, it is highly recommended. It is very difficult to maintain the kind of discipline needed within application programs. A good DBMS scheme also facilitates user report writing and easier portability, maintenance, applications development and enhancement of the system.

  This is becoming much more complex now that technology is moving towards distributed databases, which may cross different machines, systems software and suppliers. Relational models are now feasible for production environments.

- **Simple to maintain, enhance -** Employment of a DBMS helps here. Logical, structured, methodical design and use of productive development tools will also reduce time and costs and facilitate maintenance and enhancements. Good documentation is a must for this as well.

  "Object-Oriented" Design (OOD) of software also helps. Most suppliers claim to have it or at least be working on it. Compatibility with industry standards for data exchange and interoperability is very important.

- **Parameter and table-driven** - The ability to tailor the software without changing the programs, but by setting flags and values within tables in the system has proven to be an effective technique for reducing installation costs and time. Unfortunately, some software is so complex, that there are up to 6000 table entries to set just to configure a system, which is a potential cause of software bugs and implementation errors.

- **Artificial intelligence** - Or Expert Systems, as the more popular and successful branch of this technology is known, is becoming a more important factor in applications software. Even parameters, as described above, have their limitations. ES are already providing the users of some systems the ability to build/modify decision rules, criteria, algorithms and continually improve them as models of the business and as decision support tools.

- **Operationally simple** - wherever possible, the system should have the ability to be operated through simple, logical screens and instructions.

- **Workable software releases**, library/version control schemes for all software and documentation. A subject unto itself.

- **Networking** is expected today. Not just internally, but intergalactically, through support of the most common networking software, as well as Internet and "Intranet" protocols. This is such a rapidly changing situation that even the

terms just mentioned may become quaint and obsolete by the time you read this.

- **"Agents, " "triggers," "stored procedures"** and other such gadgets are designed to extend the power of modern software by automatically scanning the database and looking for certain conditions before performing specified functions such as releasing an order, creating a report, sending a message, etc., then notifying someone or even taking action, unbidden.

- **Standardized interfaces/integration** to popular software and equipment, such as APS, ERP, SCM, CAD/CAM, EDI, sensors, CNC, material handling/storage devices, text/document management, electronic mail, applications such as accounting, maintenance, order processing, customer service, SPC and others.

IV. Testing

On-line, interactive systems confound traditional systems selection methodology. Various "friendliness" issues and "software personality" are much easier to evaluate and understand in an on-line simulation mode than through documentation review and sales pitches. **A good, abbreviated mini-conference room pilot scheme is the right way to evaluate when you have done everything else and are now down to the "short strokes."** This is often called a “proof of concept” study. The methodology described here is thorough, but slow and resource-intensive. Don't do this until you think you have found the system that you want. This is not something that you do with eight or ten systems, but maybe one or two at most. **How to do it:**

1. Take those policies/procedures that you so lovingly prepared earlier and gather them all together.

2. Assuming that you have already educated project personnel in their respective functions and they have learned enough about other departments to possess a general understanding and healthy respect, you are now ready for some specifics . . .

3. Get help from the supplier to familiarize team members with the system. You probably won’t be able to do this on your own, especially on short notice. A system integrator or consultant already familiar with the system can speed things up.

4. Assign each team member part of the system, to go through a complete simulation of running the company on the software, using your policies, procedures and samples of actual or closely simulated data. Members should carefully study and learn applicable portions of the system.

5. Prepare a team book with the pilot program objectives, schedules, procedures and other needed facts. Add each member's planned agenda and test data as it is made available. Distribute it before the sessions so that others have a chance to think about it in advance and interact with each other to improve the sessions to solve anticipated problems.

6. Each team member then runs through the system what other team members present at the exercise. Problems are solved interactively at the session, using the actual software, trying out alternate methods of solving problems in different ways, altering procedures, etc. Recommendations are made "on-the-fly," but formal changes are made later when cooler heads prevail and involved managers can approve. Don't be afraid to let the vendor help. But, don't let him "stage manage" or "railroad" the proceedings. Stay in control. Keep a list of issues to review with vendors, consultants and the team.

7. During the sessions, "life size" flow charts are done right on the wall, using sample documents, Flow arrows and index cards or stickies pointing out problems, procedures or special notes for items requiring attention. Not only is this an efficient, fun way to do the job, but it makes the results of the team's work highly visible for "PR" and communication purposes as well. The results of this exercise are:

- A practical understanding of how the system actually works
- A knowledge of the strengths and weaknesses of the software
- A workable plan for the use of the system in your environment
- Identification of policy or procedure weaknesses and areas that may have to change as a result of the system implementation
- An excellent education tool in manufacturing systems

These pilot sessions are the single most effective way I know of to truly learn and understand a system for evaluation, education and implementation planning purposes.

V. SUMMARY

Now, you're ready to make the final selection and go to *contract* with your chosen supplier, which is the subject of another presentation of its own (the struggle isn't *quite* over). The contract is the key to the formal relationship with the supplier. It is especially important when anything other than the standard software is involved and when enhancements, customization or other services are required.

Major software suites usually require cooperation of many parts of an organization for success, so make sure you have built a consensus.

In conclusion, the following points about selection should be stressed:

- Most of a business system project is not software at all. Other project tasks have a profound effect on selection and ultimate project success.

- Significant changes in recent years call for changes in the way the selection game must be played for success.

- Much of other work can and should be done before and during the selection process-- education, project planning, requirements definition, etc.

- Do your planning and cost/benefit analysis in advance. Iteratively improve it as the project progresses.

- Doing all of the above is more cost-effective dollar-for-dollar than just software. It simplifies selection and keeps the process on track.

- Don't get bogged down in the function/features trap.

- Go through the 11 important supplier qualifiers first.

- Understand what motivates software suppliers and plan your strategy accordingly.

- Use the proof-of-concept testing approach to complete the evaluation and introduce the system to your firm.

Happy Hunting!
George J. Miller, CFPIM

About the Author ...

George J. Miller, CFPIM, is President of *PROACTION*, an Oak Park, CA, USA business management consulting and education firm www.proaction.net. Miller has worked with dozens of companies in assignments involving productivity, quality and service improvement, business systems, change management, acquisitions, divestitures, expert witness testimony, and others. Prior to founding *PROACTION* in 1986, he was Vice President of Marketing for Western Data Systems; Director of Planning and Development and Assistant Director-Operations for Purolator Technologies (PTI); Consultant for Booz-Allen & Hamilton, and Manufacturing Systems Manager for Becton-Dickinson. Contact at 818-706-2200 or Gproaction @aol.com.

updated 000728-- last page