Selecting Application Software and Software Suppliers

Steven J. Dennis
Smith, Dennis & Gaylord

We begin by looking at some of the data regarding the assumptions that have evolved regarding standard software packages and the software package industry.

THE MYTHS

Following are some of the myths — beliefs ("beliefs" are assumptions which may or may not reflect reality) — which have evolved over the past decade or so of application packages evolution:

- *Myth:* We can safely go ahead with our hardware purchase since there MUST be plenty of good software systems available.
- *Fact:* There are surprisingly few firms nationally that have developed truly viable packages. And fewer still that will be able to meet YOUR requirements.

In fact of the literally, tens of thousands of software firms, barely a dozen have sales of more than $10,000,000 annually. And of these, the largest barely tops $35 million . . . hardly international giants!

- *Myth:* Since all packages are essentially the same, we will budget for the low priced one . . . that will help keep the costs low.
- *Fact:* Costs certainly aren't the most accurate indicator of how good the software is . . . but you don't typically put retread tires on a brand-new Mercedes. Quality software — a system which has the quality you would expect help manage your organization effectively — is going to be a little higher priced.

There are lots of factors which go into software price — and probably only about 25% of them have anything to do with writing the programs (more on this in Part III).

- *Myth:* One great thing about a package is that we can install quickly . . . get it running in a month . . . maybe get several packages running in a month or so. . . .
- *Fact:* For one thing, remember that quality software firms are busy too . . . their schedules may not fit exactly with yours.

For another *YOU have to learn the package BEFORE your users do. You should run parallel (or at least develop a good test case). Your objective should NOT be just to see if the package works – but does it work for YOU.*

Take your time . . . do it right! After all, your organization will probably spend many months deciding on the right hardware — surely you can allow an extra month or two to make sure the software is operating properly.

- *Myth:* Our organization can't be TOO different from everyone else — we should be able to fit into a STANDARD General Ledger . . . STANDARD Accounts Receivable . . . a STANDARD Order Management System . . . a STANDARD . . .
- *Fact:* Data Processing should work for YOU . . . not the other around. Companies ARE different. Packages ARE different. Approaches ARE different. Features ARE different . . . and some are critical. A large multi-national firm doesn't just go changing its chart of accounts because THAT'S THE WAY THE SYSTEM WORKS!!!!!!!!
- *Myth:* Most applications are easy, straight-forward systems.
- *Fact:* Anyone who feels this way needs to design and implement from scratch just ONE Payroll system . . . just one INTEGRATED General Ledger to realize that there are NO easy applications.

If you find yourself saying to the software or hardware firms you are talking to, "We just need a standard A/P system . . ." catch yourself and re-think. You may be identifying yourself as the classic "easy mark." And easy marks have a way of giving their money and time to people who give little in return.

- *Myth:* The terms General Ledger, Accounts Receivable, Accounts Payable, Purchasing, Materials Handling, Resources Requirements Planning, Financial Modeling, and Inventory Control are universal . . . they mean the same thing to everyone.
- *Fact:* While your technical staff may think that General Ledge was probably just promoted from Colonel, ONLY your financial staff will be able to determine the features that are needed. Be careful . . . don't assume anything.

General Ledger does NOT necessarily include financial statements. Accounts Receivable systems don't always let you apply cash to ANY account (not just to

---

This paper is an excerpt from the book *A Success Plan . . . for Software/Implementation* by Steven J. Dennis and Barry Barnes. Published by Barry Barnes & Company, 1982.

an open receivable). Order Management differs radically for manufacturers vs. distributors.

Many systems do not allow for much FLEXIBILITY in your chart of accounts. Few packages allow you to customize the software to your organizations's own UNIQUE requirements (short of an almost complete rewrite of the software).

- *Myth:* A package is a system that is operating successfully at some other company . . .
- *Fact:* Wrong! Packages are WRITTEN to be PACKAGES!!! They are not — NOT — custom systems which happen to work at an organization which may be in the same industry as yours.

And, remember a package is also *not* something that is SCHEDULED to be done . . . it is something that IS done.

A TRUE software package is one *developed to be a package* by a firm that has *as its business* developing and supporting PACKAGES!

- *Myth:* All software packages obviously contain the proper audit trails and accountancy . . .
- *Fact:* Make sure your controller, chief financial officer, administrator, vice president of finance, business manager, or your CPA takes a good look at the package . . . we have heard of too many audits that insisted on changes to existing software . . . expensive changes . . . to include fundamental audit trails (the ones everybody *assumed* were there in the first place).

The people who write software don't always understand accountancy considerations . . . on the other hand, just because the firm has strong accounting credentials, doesn't mean that their software adheres. ASK about controls, audit trails, security, and the philosphical underpining of the software products.

- *Myth:* Just because you have a computer or are getting a computer . . . just because management feels that EVERYTHING should be automated . . . just because there are relatively inexpensive packages crying to be bought, we should surely bring ALL applications in-house.
- *Fact:* Some applications require careful analysis before a final decision is made to go in-house. Payroll is the best example: For small companies we have often recommended against the hassels of maintaining their own payroll . . . changes in tables, government forms, minimum wages, unions, etc., require a heavy in-house investment . . . well worth it for many organizations — but definitely not for everyone . . .

Be appropriate. Automate when there's some definable distinct advantage to the organization. Automate when you expect to be able to see *results*.

- *Myth:* For those of you whose organizations have an in-house data processing staff, "there just isn't ANYONE out there who can develop a system better than we can right here in our own organization" (also known affectionately as the "Not Invented Here" syndrome . . . or, often, more accurately, as the "Kiss of Death" or "Results Not Yet In" syndrome).
- *Fact:* First off, you are probably viewing it from the *technical* side . . . and from that viewpoint there may well be some truth. After all who better knows the DP philosophy, particular hardware configuration, internal politics, etc., better than your own data processing department.

In fact, the software house you select should be an expert in PARTICULAR applications — they know General Ledger, Order Management, Payroll, Medical Billing, Financial Modeling . . . and in the long run THAT's what you need.

And, by the way, (to the surprise of the DP staff) a good software package will often be impressive technically as well). This is much truer now than in the past. "Mature" packages are often relatively new — and often written using software development tools that simply weren't available a few years ago.

- *Myth:* With a wealth of new systems and languages, programming is now much easier than before . . . surely I can write my own applications.
- *Fact:* Programming is not so terribly difficult . . . but desig a particular function — or an entire system — requires the experts. The main benefit of the advent of powerful software development tools is that they free up time to do a more comprehensive job of DESIGN and that's fundamentally why packages are so suddenly such a viable alternative.

Learning to speak another language is one thing . . . writing a novel using this new language is quite another! And, writing that novel error-free? . . .

Programming is only approxiamtely one-sixth of the total effort . . . the whole picture looks something like:

- Design ..... one-third
- Programming ..... one-sixth
- Test/Debug ..... one-third
- Training & Documentation ... one-sixth

- *Myth:* We MUST have our programs written in COBOL, or some other such language.
- *Fact:* Arguing for a particular computer language is usually ridiculous. It's like arguing for French, Spanish or German — all of which are excellent languages.

Properly used — FORTRAN, RPG, COBOL, BASIC, and many other user-oriented high-level languages can provide *excellent* solutions.

Remember, even though *you* may think English or Danish, or whatever is the world's best language, millions speak others . . . write others . . . get results in others.

- *Myth:* Choice of language doesn't matter AT ALL . . . choice of file handling technique doesn't matter AT ALL . . . as long as it WORKS!!!!
- *Fact:* Buying something completely non-standard

can be a disaster . . . insist on complete documentation for anything that looks a little out of the ordinary.

• *Myth:* Since my company deals with a single-person (or small) law firm . . . or a single-person or small CPA firm, it's okay to get my software from a very small software house . . . or even from an individual.

• *Fact:* This is a tough one . . . certainly there are many, many excellent, well-qualified software firms. Remember, though, our industry does not yet have a bar exam or any accepted professional certification like the CPA . . . nor are there ANY levels of standards throughout the industry which compare with the standard "generally accepted" accounting practices that all CPA's follow.

On the other hand, if your small (one, two . . . five-person firm) KNOWS their business — and yours — they *may well be far superior* to the 100-person company which views you as a small fish in a big pond. We know of a company — one of the 10 largest in the world — which actually PREFERS to work with small (one to ten-person) software firms.

Put this test to work . . . if my CPA went out of business, where would I be? Probably all right — there are others who could step right in. NOW if my software consultant/supplier went out of business then what???

• *Myth:* There are software firms to whom I can turn over complete responsibility for my implementation . . . total *turnkey* solutions . . . software houses which will sign a contract guaranteeing success . . . with penalty clauses . . .

• *Fact:* There ARE firms which will tell you that they'll take on all responsibilities. But let's face it . . . whose system is this? Theirs? Unless *you* take the responsibility for the solution to work . . . invest the time . . . invest the energy . . . adopt the right attitude . . . you will be developing all the ingredients for failure.

And, remember, desperate people will sign *anything*. A firm that knows how to do business doesn't have to sign your attorney's document in blood — they'll drop you like a hot potato and move on to someone else who knows how to get results.

• *Myth:* Custom software is never necessary or if it is, it should always be done in-house.

• *Fact:* Not true! Custom software IS quite often required.

For example: We have worked with a large client which fabricates and erects steel for many of the really *large*, modern high-rise office buildings and hotels in the Western U.S.A. Recently, a custom system was developed (by an outside firm) for this steel fabricator — one that estimates, to the nut and bolt level, the steel requirements for a 40-story office building . . . determines the best source throughout the world for that steel . . . how to transport it . . . and how long it'll take . . . and cost. That doesn't exactly lend itself to a package.

SUMMARY

There is an emerging — definite — context for standard application software packges. The degree to which workable solutions and procedures evolve for the successful incorporation of this new field into our business life directly affects the results we can expect for the near future.

The use of the data outlined in this presentation can assist in *initiating* the process of getting RESULTS . . . for your organization and for others.

Now, let's move on to an examination of the process of assessing software, selecting the software supplier, and implementing the software system.

PART II
SELECTING A SOFTWARE SUPPLIER

INTRODUCTION

The approach outlined in this section of this book is most appropriate for companies in the $10,000,000 to $10 Billion annual revenues categories. Smaller companies tend to be able to fit extremely well into totally "standard" packages — often being able to change their mode of operation to fit the package. Larger organizations — particularly when they hit to $25,000,000/year level — tend to have developed unique operating styles, unbendable procedures, inflexible managerial requirements, or just plain strong preferences.

In general, very small organizations can ignore a lot of what we propose in this book. Yet . . . the fundamental underpinning of RESPONSIBILITY for results — a commitment to being successful — will still serve well!

CONTEXT — A VARIETY OF VARIABLES

Packaged software has become the major area of focus in the information systems field. Yet, the industry called "Packaged Software Firms" has no equivalent of a FORTUNE 500. In fact, only recently has the situation arisen where there are more than a dozen companies in the world which have annualized sales of more than $10 million (and those few have only recently attained that level).

The vast majority of software firms — including the QUALITY ones — are small companies doing between one-half million and five or six million dollars per annum. And, small businesses are sometimes subject to radical ups and downs.

Thus, the selection of a software firm needs to be based on a set of criteria which optimize the potential for success. It may well be the case that the *firm* supplying the software is as important — or more so — than the software itself.

The important point to know . . . and acknowledge . . . and accept — whe you like it or not — is this: In selecting a software supplier, you are establishing a *long-term business relationship!* And, if you do your job WELL, you'll establish a really long-term relationship.

So DO your job well. Exercise the same care you'd use in selecting your CPA firm and your Corporate attorneys.

PURPOSE — A TOOL FOR MEASUREMENT

This set of guidelines encompasses what we've learned from our own experience as consultants and as software suppliers. It is developed from a variety of viewpoints. We offer it as a tool to use to measure any software firm which offers standard application software packages.

GUIDELINES — CHECKLISTS & PROCEDURES

Following are a series of "bullet-item" guidelines. These can be greatly expanded: These lists are by no means meant to be all-inclusive . . . they're simply a good start.

Give the software suppliers you deal with an opportunity to present their story to you — in their own way; then use these items as a checklist.

Don't expect any one firm to get an A+ on all items. We *know* that we're not "There" on them all . . . and we probably never will be! A score of 70 to 80% is "Excellent"; 90%, "Superior"; 100% . . . well . . . come on . . .

THE SELECTION COMMITTEE

Application software packages should NEVER be purchased by a single person . . . and that's not an indictment that *individuals* can't adequately make the decision. Some can.

The truth is: Software must be implemented by a variety of people at different levels in different functions within the organization. The wise organization will get a variety of viewpoints from the start.

In general, the Committee Model doesn't work in this world (as governments strive ernestly — and repeatedly — to prove). Yet, here is an example where a team of well-chosen effective people can make a real significant contribution.

Ideally, the Selection Committee should include representatives of the following functions . . .

- Organizational Management
- Functional Management
- User/Operator
- Data Processing/Technical
- System Implementor — the person who will have the responsibili of successfully implementing the system once it's chosen.
- System Coordinator.

A properly selected (and operating) Selection Committee can achieve tremendous results for the organization. It should meet frequently during the buying cycle. Each member should diligently fulfill assigned responsibilities.

THE SOFTWARE/SYSTEM ACQUISITION PROCESS OVERVIEW

We can't say what works for everybody . . . but in our experience as managers, users, consultants and software suppliers, we've found the following procedure to be a valid one:

Organizing Yourselves

- Define the Selection Committee
- Define the overall, broadbrush implementational schedule
- Develop a *brief* Selection Committee charge and guidelin
- Develop a *brief* background and requirements document fo the software suppliers

The Review Process

- Define the software packages to evaluate
- Poll your colleagues for additional ones
- Call your friends
- Ask around at social occasions and cocktail parties
- Poll the computer vendors for others
- Contact your CPA (and other consultants)
- Preview the various directories
- Define acceptable computer vendors
- Collect literature & documentation
- Contact the software suppliers
- Contact the hardware vendors

Our recommendation is an unusual one. We recommend that you do the front-end work yourself (previewing brochures, telephoning software suppliers, calling a few of their resources), and quickly narrow it down to three to five finalists which you'll then *visit* . . . and then send your RFP (if you use one) to only those. It saves you a lot of time and energy in the long run, while letting you be sure that the supplier who responds with a proposal actually knows something about your business.

The Preliminary Evaluation Process

- Develop a preliminary set of selection criteria for the software packages (Software Requirements Checklist)
- Develop a preliminary set of selection criteria for the software firm
- Find out more about the software firm (by telephone)
- Find out more about the application packages
- Procure the software supplier's client lists
- Telephone the software firm's references
- Select three to five (3-5) finalist firms
- Visit the finalist software firms
- Visit the computer vendors

The Interim Evaluation Process

- Analyze the findings of the visits
- Expand the Software Requirements Checklist
- Adopt the budget for software, hardware, etc.
- Develop the Request for Proposals (if appropriate)

The Final Evaluation Process

- Receive and review the proposals
- Prepare the Comparative Requirements Analysis
- Review the responses (entire Selection Committee)
- Review the responses with your users
- Review the responses with the software suppliers
- Select and advise the software supplier

The next step may be the most important part of the process. Make s you get to know ALL the people with whom you'll be working . . . and that they get to know — and like — all your key people. This is a people-oriented business. The better you know, and understand each other, the better will be the overall level of affinity and communication and reality when the going gets tough . . . and it *will*, at one time or another, get tough!

Establishing the Client/Software Firm Relationship

- Complete the financial requirements & procedures
- Complete the legal/contractual requirements
- Establish the technical support contact points
- Establish the software training procedures
- Establish the documentation update procedures
- Establish the user support procedures

In general, it's the hardware vendor who will maintain the computer equipment and the operating system. DON'T rely on the sales representative . . . he or she has other sales to bring in! Get to know the people who will support you.

Establishing the Hardware Vendor Relationship

- Establish the relationship with the hardware vendor's operating software and hardware maintenance staff (SE's & CE's)
- Complete the legal and financial requirements with the hardware vendor
- Issue a purchase order for the required computer hardware and operating software

Implementation Planning

- Define the Implementation Review Committee (may be the same as the Selection Committee)
- Adopt a requested Implementation schedule
- Present the requested schedule to the software supplier for review and resolution
- Review the software supplier's recommended Implementation Plan
- Mutually resolve discrepancies to achieve an Adopted Implementation Plan

General Application Preparation

- Schedule applications training
- Schedule technical training (if appropriate)
- Review procedures for potential modification
- Conduct weekly or bi-weekly Implementation Committee review sessions
- Procure the software supplier's final recommendation of the hardware configuration
- Finalize the hardware configuration

General Computer Hardware Preparation

- Conduct the site review for the computer
- Prepare the computer site as appropriate
- Analyze & define CRT and hardcopy printer locations
- Arrange for cabling and modem installation
- Conduct Implementation Committee review meetings

Final Implementation Preparations

- Initiate applications training
- Initiate hardware training
- Initiate other technical training
- Conduct project activities for special/custom work
- Conduct Implementation Committee review meeting

Implementation

- Install the computer
- Install the software module(s)
- Load/convert data to the new software
- Conduct application testing
- Conduct Implementation Committee review meeting
- Conduct end-user operational training
- Implement the application(s) on a "live" basis

On-going Review

- Conduct periodic review sessions with end-users
- Conduct periodic Implementation Committee review meetings
- Conduct periodic software firm review meetings
- Conduct periodic hardware vendor review meetings

This checklist is *one* way to acquire software. It's surely not the ONLY one . . . it just works! Try it . . . or modify it. But whatever you do, have a commitment to get results; develop a a Plan . . . then WORK the plan.

THE SOFTWARE REQUIREMENTS CHECKLIST (THE STATEMENT OF REQUIREMENTS)

The Software Requirements Checklist (also known as the Statement of Requirements) is the common thread for the *Functionality* of the software to be chosen. Most of what should go into this document is a list of *features* which you want or need . . . thus, it's impossible at this time to descri exactly what should be on it. Following, though, is a set of guidelines for preparing it:

- *Don't overlook the obvious* . . . don't "assume" that what want will automatically be in all packages.
- *Clearly state your requirements.* Avoid lots of text . . . it doesn't get read; use lists or checklists, where possible.
- *Rank your requirements.* Don't get too fancy . . . "A" for Must-haves; "B" or "3" for Highly-Desirable; "C" or "2" for Nice-to-Haves or Tie breakers.
- *Include philosophical or approach requirements* — things like on-line vs. batch . . . language . . . options . . . decentralized vs. centralized management style . . . growth . . . plans . . . security requirements . . . accounting batches . . . auditability . . .
- *Include interface or customization requirements*
- *Ask about product expansion* — are updates provided? How How often? What happens if you customize? Interface? Is a software support agreement available?
- *Ask about documentation* — what's provided? How readable is it? Who writes it? How often is it updated? Do you *get* it? How many levels of documentation are there?
- *Ask about training* — what's provided? How often? Who attends? Are there standard classes? Do you get the training materials? Who conducts the training? Who attends the training?
- *Ask about installation procedures* — what checklists wil you have? What procedures will be provided? What assistance will you get?
- *Find out about support* — what happens when you get in trouble? How do you report software bugs? What facilities exist for phone consultation? How often is user documentation updated? What procedures exist for follow-up on your requests? How do you suggest changes and the "wouldn't it be nice . . .," enhancements or extensions? Does the firm have a standard support program and a standard support contract?
- *Describe your organization* — tell the software firm abo your objectives (lists — not narrative); get each functional unit's requirements; get alignment so that you'll all be using the same terms . . . expecting the same results.

A well-designed Software Requirements Checklist should have terse one- and two-line features/requirements statements with a place to the right (or left) for the software supplier to enter a short yes/no/"*" response. Each *grouping* of features should be followed by a "*" blank space to write responses, exceptions, n rates, quotations for custom work, future release dates, etc. In other words, make it EASY to USE.

WHAT ABOUT CUSTOMIZATION?

No matter how much you may desire otherwise, your application may well have requirements that just aren't available from a standard package. We've seen requests for such applications as railcar tracking, event scheduling, loan tracking, event-driven action item management, and such — applications which are mainstream to the company, and which must reflect the *specific* approach of the organization.

The need for customization need not be a catastrophe . . . IF . . . you understand the implications. Learn how to define the need:

- Check against the Statement of Requirements for *alternative approaches.*
- Use the software firms' *consultative assistance* — they have suggestions as to how others have solved the same problem.
- Seek out *parameter-driven software* — it may be tailorab your need.
- *Re-evaluate* your need . . . if there's no package availa just may be that you're doing things too differently.
- Don't be afraid to *stick to your requirements* . . . perh organization's way — non-standard as it is — is the one which gets results!
- Determine whether standard package modifications required a *structural or are general enhancements* . . . that is . . . are you adding a room . . . or repainting . . . do you need a new foundation. IF the changes are structural — go custom (or buy the package to use as a building block for a custom system, with the understanding that you must take ownership of the resultant system).
- If you've found a "fit" or a near-fit for other applicati *discuss with the software firm* their interest in developing your custom requirements . . . or their recommendations . . . or how cooperatively they'd work with another firm which you might engage.

Most software requirements can be met from a package . . . but NOT ALL . . . not even all the "of course THAT would be in a standard package" requirements for a given function may be in the package you like best (and it may still be the best package to buy).

THE SOFTWARE DEMONSTRATION

Seeing the software work is extremely important. Software — like the people who design and use it — has a personality. Ask for a software demonstration!

The demonstration is best done at the software firm itself (assuming it has its own computer).

- Define in advance which modules you want to see.
- Send the software firm a copy of your Statement of Requirem in advance.
- Get the RIGHT people there.
- Come prepared.
- Give the software firm an extra hour or two to tell their story.
- Take the software firm's advice.
- Allow sufficient time.
- Ask for an extra hour or so — at the end of the session to unanswered questions handled.
- Tell what you expect to accomplish.
- Keep an Open Mind!
- Keep on Track.
- Be courteous.
- Ask Questions.
- Look at the Audit Trails.
- Look at the User Documentation.
- Look at the Technical Documentation.
- Look at the human engineering.
- Expect to have a reasonably good fit — (75% to 85% is considered a good fit; 90% is considered excellent; 100 is rare!).
- Expect to find gaps.
- Beware of the Everything's Great Syndrome.
- Be professional.
- Meet the user support staff.
- Don't demand to spend lots of time with the technical staff
- Follow up on Unanswered questions.

A good software demonstration can enlighten you as to omitted items on your Statement of Requirements. Also, it's an opportunity to interact with key people in the software firm who may end up being your contacts for years to come.

Use the demo effectively . . . then go back home and update your Statement of Requirements. *Have a Selection Committee meeting after each such visit.*

THE REQUEST FOR PROPOSALS

This is probably the area in which the *most* mistakes are made . . . by the requestor!

In the first place, use an RFP where it's *appropriate*. If you find a package during the preliminary search which meets your needs . . . offered by a firm that fits all the selection criteria, then, WHY go through the misery of an RFP? And, the RFP is as much work for you as for the software firm (if you're doing it properly).

Admittedly, it is a controversial stance to recommend *against* RFP's . . . but more TIME — and, often, MONEY — is spent by some prospective buyers, going through the drudgery (and motions) of Proposal Requests than is often spent on *acquiring and implementing* the software itself!

Let's look in more detail at the *purpose* of the Request for Proposals.

What an RFP Isn't

- It is NOT a way of justifying an already-made decision.
- It isn't a guarantee (those are up to YOU!)
- It isn't a fancy way of covering shoddy selection processes.
- It isn't a "test" for the software supplier (quality soft firms throw away the ones which look like "tests").
- It isn't a way of badgering others into doing things your way.
- It isn't a document more than ⅛" to ½" thick (anything thicker is a request for free consulting services).
- It is NOT something to be tied into a contract (most software firms which will agree to tie the RFP and their response to the contract aren't capable of living up to a court challenge; who *can* deliver what they say aren't interested in complicating their legal agreements . . . on strong counsel of their own attorneys!).
- It isn't, in summary, much more than a statement of requirements . . . than a statement of requirements . . . a Statement of Requirements.

What an RFP IS

- A Statement of Requirements.
- An opportunity for the software firm to tell its story (in *own* way).
- A place for summarization of costs.
- A place for summarization of potential implementation sched dates and events.

Guidelines for the RFP

- Make it *friendly!*
- Make sure you *personally* contact the firms you send it REPEAT: Personally telephone — or, better yet, VISIT — the software suppliers you want to respond.
- Send it to NO MORE than *three to five* firms.
- Tell about your organization (briefly).
- Give a brief background of the situation for which you're seeking solutions.
- Include the Statement of Requirements.

- Allow the software firm to answer in its own style.
- Attach an outline of what you want to know about the softwa firm. Keep it to the "need to know" level.
- Avoid rigid formatting requirements.
- Don't demand all sorts of contractual modifications . . . suppliers just aren't interested in doing business that way (no matter what counsel you get otherwise).
- Don't ask the supplier to tie their response to a contract good firms spend enough time on their contractual agreements to do business well — and their attorneys counsel them heavily against such modifications.

On the other hand, if the software firm has shabby-looking or weak contracts, you're entitled to ask for contractual modifications . . . but . . . do you really want to do business with such a firm?

- Remember: The software firm is busy — particularly if they're competent. If you ask for two weeks worth of work to respond to an RFP, you'll only get the software suppliers who aren't in demand.
- Get competent assistance in evaluating the responses.
- READ and ANALYZE the responses.

As a summation, regardless of whether you agree with us on the value of an RFP, don't let the RFP be a substitute for human interaction, client reference-checking, . . . software firm visits . . . and just plain hard work. If it's worth doing, it's worth spending some time doing right.

EVALUATING THE SOFTWARE FIRM

It is almost as — maybe even more — important to carefully assess the software supplier as the package itself.

REMEMBER: You are establishing a long-term business relationship. Just because this is a highly technical field, don't be bamboozled into anything. Look for the *same* organizational attributes that you'd look for in ANY company!

The successful software organization has to know their business . . . and must be committed to supporting clients using standard software products.

Following are some checklist items we've come across. Perhaps you can extract several items and develop a weighting/evaluating schema which will work for you.

- *Look at Outward Appearances:* How does the software firm va itself. What evidence is there that they've been around for a while . . . survived the magical "New Company Mortality" syndrome? How have they invested in the future? Look for "gut-level" feeling. DO they *look* like winners? Check these attributes:
  - Facilities
  - Equipment
  - Furnishings
  - Clients
  - Checklists & Procedures
  - Documentation
  - How the people view things (values)
  - Organizational Structure
- *Look at the People who Create the Packages:* Again, use gut-level approach. Would you want them working for your organization?
- *Look at the Firm's Background:* More gut-level stuff. Lo them and their experiences . . . and don't assume there's any one "right" way. Listen carefully to their story.
- *Look at the Knowledge Level of the People:* How have the all together? Does the firm have what it takes to succeed in ALL its areas (not just the technical)?
- *Look at their Guidelines and Standards:* Examine the evi of commitment. *Written* standards are statements of intention, stability, permanence. They're important!
- *Look at their orientation:* This is incredibily important . . . this is the firm's *real* purpose. Ask candid, yet open, friendly questions.
- *Look at the Firm's Target Marketplace(s)*
- *Check for Internal Procedures:* These are the indicators attention to detail. And, that's critical in the software field.
- *Look at the Support Apparatus:* This will be your contac AFTER the sale. You only interface with the sales or business development function BEFORE you sign up . . . IF you select the right firm.
- *Look at What Goes Into Product Price:* The difference in price is immense. There are, in many cases, equally-developed software packages available from different firms at vastly differing prices. What's the difference between a $3,000 Order Processing system and a $40,000, $50,000, or even a $100,000 one? Generally, there ARE features differences . . . but not always. The difference may well be in the firm. How does a firm selling their product for ten times that of another one of equal "features" survive? Generally, because they appeal to organizations which *value* long-term commitments.
- *Look at the Software Firm's Sales Style:* Here's where a of the true values of a firm show up . . . to the extremes. If the sales representatives act like a stereotyped salesperson, then there's probably something behind the scenes which supports such an approach. On the other hand, if the people responsible for business development show good knowledge, experience, and a consultative approach which demonstrates genuine concern for your success, they're probably reflecting some

very solid and fundamental philosophies and policies of the firm . . . latch onto them.

- *Look at the Implementation Planning Assistance:* The understanding of implementational considerations is one of the most positive indicators of a real-world, results-oriented firm.

  If the firm begins talking implementation, listen to them. Chances are, if they've passed the other tests, they know far more about how to implement a system than you. After all, they're probably doing it between ten and a couple of hundred times per year!
- *Look at the Legal/Contractual Instruments:* If you get a double-spaced typewritten page for a contract, feel free to take a hatchet and an army of attorneys to it.

  On the other hand, if you get a typeset, well-structured document that provides for mutual protections and which incorporates and documents business procedures, then expect the firm to be relatively resistant to modifying it.
- *Look at the Firm's Growth:* The software industry is in explosive environment.

  Unfortunately, even organizations with shoddy products and shabby outlooks can survive — even grow rampantly! Growth is a tough thing to handle . . . and it's roughest on the firm which is committed to quality.
- *Look at the Company's Management Style:* Despite all the growth, the firms which will *truly* succeed (for themselves and for you) are the ones which MANAGE themselves well . . . just as in *any* field, good management pays off.
- *Look at the Company's Business Ethics*
- *Look at the Software Firm's References:* Ask for references . . . AND . . . then *contact* them.
- *Talk to the Software Firm* — This has to be the most imp criteria of all. Talk to the software firm as you would to ANYBODY who could truly assist you; don't worry about giving too much of yourself and your values to them . . . if they're unethical, they'll *definitely* try to take advantage of you — be mature enough to be willing to find that out beforehand.

  Choose the software firm as you would your CPA firm or corporate attorneys. Choose them using the same criteria you would if your organization were going to acquire them . . . or if you were going to invest in them.

AFTER YOU SELECT THE PACKAGE . . . THEN WHAT?

There are several steps which should be taken. The important thing is NOT to assume that you're "there" . . . indeed, the journey is still somewhat in its infancy. In fact, you're just beginning! There's some more internal organizational analysis that needs to be done . . .

- Determine what people-problems you will have with software tha cuts across organizational lines . . . Order Processing for example can affect marketing, sales, credit, customer service, production control, manufacturing, shipping, quality assurance, AND accounting . . . what will your *people* problems be?

  Develop a plan for handling the inevitable . . . it WILL occur!
- If you have to modify procedures to fit a selected package, try it *manually* first. Get the resistance out of the way . . . PRIOR to having the computer to blame.
- Get the user to sign off on the system . . . the Accounts Re supervisor will be much happier if he or she blesses the system *in advance*.
- Take ownership of the system . . . and make sure everybody — including management — expects results . . . and is committed to doing whatever it takes to GET results. Finger-pointing and blame and "reasons" just simply have no place in the implementation phase. If they crop up, acknowledge them for what they are (the things people do when they're NOT getting results) and MOVE ON! (to getting results).
- If you haven't already done it, list your required enhancem Have the software vendor quote/ recommend how these enhancements should be done.
- Develop workarounds for all the functions which aren't exac the way you'd like the package to work — and inform everybody, so there won't be the excuse of: "Well, this package just isn't the way we should be doing things."
- Make sure you get completely trained on the software (from perspectives: User . . . technical . . . and standards). Make certain the user is fully trained . . . that there's the *ability* and *willingness* to understand.
- BE PREPARED — remember that users CAN damage themselves thr no fault of the software house or the software.

  Or even: "Advised of a schedule change??? Call the dispatcher."

  Convert some or all of the members of the Selection Committee into an *Implementation Committee*.

  Identify ONE person (for each module) as the System Implementor — that one person who has the ability and the responsibility for getting results . . . and who is recognized and respected by others as able and willing to make it happen. Have a meeting of the key players at least once every two weeks.

  Once the selection has been made, there's the

cumbersome job of getting things rolling. And that's where the software firm's many experiences can assist you . . . that's where Implementation Planning and Project Control procedures come into play.

THE PURPOSES REVISITED

The purposes of this document are plain and simple: To provide at least *one* quality, honest, "what's so" step forward. Specifically, we offer a considerable amount of data, several methodologies or processes, and a sharing of experiences which may support you and your organization in attaining the successes you want. The only real value you can get from this book is a willingness to look at things *as they are* . . . followed by using whatever portions of our data, methods, and experiences which prove to be useful for you.