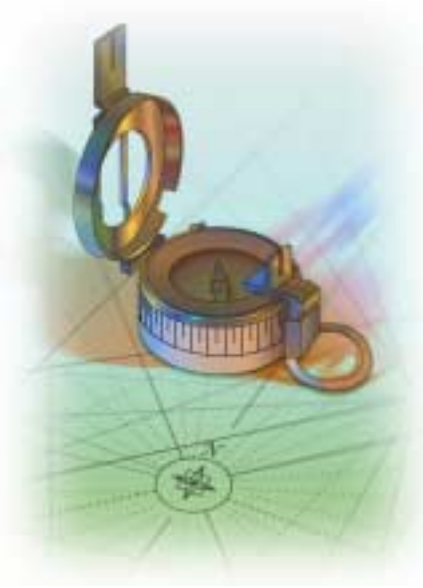
Outsourcing: The Benefits and Sacrifices



By Richard Latty
Solutions Engineering Corporation
7830 Old Georgetown Road
Bethesda, Maryland 20814
rlatty@soleng.com
www.soleng.com

Outsourcing is the strategic approach to providing a capability to your organization by having it provided by another organization - one that specializes in that specific area. This approach applies to many different capabilities and services. This paper focuses on computer software and provides a summary of the benefits and sacrifices, which accrue to an organization by adopting this approach. Outsourcing software is not a good strategy for every organization. But where it fits, it is an exceptionally good business strategy. There are at least two ways to view the issue - one from the viewpoint of a software engineer, the other from the viewpoint of management of a company with information system requirements. We will look at both.

The benefits of outsourcing to the client include benefits enjoyed due to aspects of the software engineering company and due to the client not having programmer and development staff in-house.

Benefits from properties of the outsource software engineering firm include:

- Breadth and Depth of Team,
- Collective experience of the Team,
- Professionals with repeated Full Systems Life Cycle experience.
- Structured approach and its cost effectiveness,
- Efficiencies of previously written code,
- Knowledge of end product capabilities from the beginning,
- Knowledge of timeliness throughout by milestone definition,
- Tools available,
- Most cost effective use of costly talent (provided by Breadth of Team),
- Definable costs for product enhancements,
- Implementation tools and technologies selected to fit the needs of the project, not individual programmer skills,
- Configuration, Training, and Help Desk services.

Benefits from avoiding in-house programming staff:

- Reduced payroll and fixed expenses,
- Freed Hostage - No dependence on employee continuity for software support and maintenance,
- Only scheduled, tested, and approved releases are provided to the user community,
- Overall reduced costs, and no on-going developer costs associated with use of the software,
- Ability to evaluate cost/benefit of any proposed enhancements,
- Decisions made in the best interests of the users and core business of the company - not curtailed by programmer issues or sensitivities,
- Reduced time of staff consumed in design, development, testing, training, and documentation,
- Reduced time required to deliver product.

A more detailed discussion of each of these follows.

II. BENEFITS FROM UTILIZING A SOFTWARE ENGINEERING FIRM

Breadth and Depth of Team. Outsourcing will give the company a breadth of team members that are rarely provided by a typical in-house staff. To build sound, versatile, and comprehensive software, design databases which accommodate near and long term needs, you need capable, experienced, and focused talent. High quality information systems are not just written. Each step in a series must be completed correctly in order to provide a high quality, relevant, and durable IS (Information System). This is accomplished by an IS (or "System Development") team. The team is made up of people with different talents, training, interests and experience. The collection of people which are typically available within a Software Engineering firm include: Systems Analysts, Data Analysts, Applications Specialists, Design Engineers, Software Engineers, Database Engineers, Programmers, Quality Assurance & Test Engineers/Technicians, Technical Writers, Trainers, Network Engineers, Database Administrators, Program Managers, Contract Managers, Web Designers, Graphic Artists, Help Desk Techs, and others. Some of these will appear in a single individual, but each role will have several people whose primary role (and expertise) is of that position.

The Software Development process requires multiple talents and expertise found in teams of individuals. Management and operations staff must be queried to obtain the full set of Requirements, priorities, constraints, views of the future, and inputs relevant to the design of the IS. Design team members compile these into a Design Definition and iterate with management and operations staff in defining the target IS (or the blue print of the IS) to be constructed. The software development team members build the IS to the specification. Every hour spent in the design definition will save many hours in the software development effort. Additionally, it will save hundreds of hours in time required to support the IS later on. A well designed IS will differ in many significant ways from a pre-maturely coded system. The well designed system will:

- require less code to implement,
- employ existing code to a greater degree,
- be more modular,

- have common functionality achieved by the same code bodies,
- be buildable in fewer iterations (have fewer Builds and, therefore, fewer exceptions and test cycles),
- take less man hours and calendar time to deliver,
- have a more versatile data architecture which supports extensions and add-ons,
- clearly meet the stated Requirements (through a Design Audit),
- have well-defined systems documentation *before* code development,
- be less expensive to deliver and use.

As the software development team is building modules to the IS, the Quality Assurance team is confirming that the software adheres to the specifications set forth in the design, and the Documentation team is translating the design specification and the operations of the software into user documentation and systems documentation. Rarely can these different roles be filled effectively by one person each. Each of these is a very important element of constructing a successful IS. Neglecting any of these elements will result in limitations of the target system. Outsourcing should give you access to personnel of much greater technical depth, experience and training than you would have on your internal staff. Many outsourcing companies will have senior software, database, communications, or network engineers with advanced degrees (Ph.D. or M.S. in Computer Science or Electrical Engineering, MCSD, MCSE, Certified Network Engineer, or other certifications). While they are not generally assigned full-time to a single IS development project, the team members do have access to this talent and intelligence when needed to address particular IS issues. Team members of this caliber migrate to firms that develop software for client companies and organizations. They need a steady diet of challenging, interesting, and worthy projects to keep them employed.

There are some types of projects that can be successfully completed without this breadth of talent, but these tend to be small in size and very limited in utility and impact on the company profitability. The project will greatly benefit from Breadth and Depth of Team if the project:

- Supports/Automates numerous or complex business rules,
- Employs multiple technologies working co-operatively,
- Requires sophisticated, complex, or large data architecture,
- Would benefit from software re-use,
- Employs a nascent or newer technology,
- Interfaces with external systems,
- Has team members who have experience with similar projects in the past.

The breadth and depth of the development team provided by the client “in-house” is very rarely comparable to that of the professional software development firm. However, the larger the number and scope of projects performed by the “in-house” staff, the less this is true.

Collective Experience of the Team. There is nothing that compares with collective experience or collective intelligence. If our knowledge base could only benefit from the experience of a couple dozen of our professional peers, we would be vastly more effective. Outsourcing enables you to tap into the knowledge and experience of professionals who have addressed IS problems very similar to your own. The value of this knowledge and experience would be difficult to over-state. They are the basis for: quickly arriving at workable designs, providing accurate estimates of time to deliver and various personnel time required, tool selection based on the nature of the project, which existing code has a good fit and dramatic efficiencies gained from re-engineering existing code, ..., etc. This industry is replete with examples where qualified personnel have spent weeks or even months grappling with a problem, only to have the solution identified in hours or minutes by someone brought in who had more experience in that particular problem domain. The axiom, "Knowledge is Power" applies to this field like no other. If it could be stated more accurately, it would be "Knowledge is Time and Money Saved". The value and cost reduction affect of the collective knowledge and experience alone, could justify outsourcing a development project.

Structured approach. No one would think of building a house by hiring a carpenter, have them go to the site, discuss what is wanted for a house with the future resident, and then just start building the house! The amount of in-house software that is constructed in this fashion is mind-boggling. Members of the design team must spend the needed time with the pertinent members of the client organization in order to accurately and completely identify the needs of the organization and their priorities before **any** code is written. It is here, with the design team members of the software engineering firm that the personnel resources of the organization should be focused. The quality of the job performed in the design phase will define the potential quality of the end product. No subsequent phase of the development cycle can make up for or recover from deficiencies introduced in the design phase. The importance of this phase is highlighted because this phase conventionally is neglected or hastened by in-house programmers. "Heads down programmers" are an essential element of the development team. They are necessary, but alone not sufficient, for a successful development project. Outsourcing gives the client's project the benefit of ***all members*** of the development team who, through their work as a team, greatly enhances the quality of the product.

The structured approach includes: accurate and complete definition of the problem or task; thorough documentation and understanding of the company business rules and workflows, complete and consistent system design (complete definition of the data structures, user interface, work flow, performance requirements, multi-user considerations, security, error sensitivity, recovery, support utilities, inputs/outputs); well engineered implementation (database definition, communications engineering, software requirements definition, library selections, identified re-engineered components, and code development); quality assurance and testing (verification of the design as well as operational modules); documentation (user and systems); training procedures (where needed); end user technical support or "help desk" provisions.

Efficiencies of previously written code. This topic covers two closely related subjects: 1)employing existing modules unchanged (libraries of selected functionality), 2)re-engineering existing modules to extend their functionality to meet the needs of a project. Comprehensive and well tested libraries of routines are now available for just about any topic in development ranging from sound, imaging, communications, database, data entry, graphical user interface, on-line and context sensitive help, bar code, and others. Many of these products are accompanied by suites of development tools, support utilities, migration utilities, drivers and testing tools. This has become a very hot topic now that the development community has accumulated a very considerable body of software. The re-usability of this software is greatly increased by the fact that it was written with the object oriented design paradigm (or has been written in a highly structured and modular fashion). The world of software development has created literally millions of components. Many of these are well-designed, well-constructed, field-tested, well documented, and fully amortized components. Application software is increasingly becoming an integration of previously written components. The software developed or written for the system is increasingly the “glue” that connects these pre-existing components together. The result is better, cheaper, faster to develop software. This is especially true when the cost of the libraries can be amortized over many projects

Software Engineering firms are quickly accumulating these components into their tool chest. These include components constructed for client projects as well as third party, purchased libraries. Many times new employees will bring a transfusion of knowledge regarding new and useful components with them. Systems interfaces are excellent examples of in-house developed components with high re-usability. For example, a systems interface built to support services to the Federal Express API, a credit card services API, or SMTP (Simple Mail Transport Protocol) services would have extremely high re-usability for other applications requiring similar services. Compare this to the time and effort required to write this (and almost everything else) for the first time. The value of this experience and “tool chest” becomes very clear.

Any software engineered for ease of maintenance and modular functionality is a high candidate for use in re-engineering. For example, a subroutine used to determine the weekday of the first day of a month and year, could readily be re-engineered to determine the weekday of any day-month-year combination. The increased size of the pool of existing software and the experience in re-engineering existing code can make outsourcing extremely cost effective. The time required to re-engineer an existing code body may take 5 to 10% of the time required to write

the same code body from scratch. The result is huge savings to the client. Additionally, re-engineered code is much cheaper to support, refine, and enhance than is new code.

Knowledge of end product capabilities from the beginning. Working with an experienced software development team provides the client with a very good understanding of the expected capabilities of the end product very early in the development cycle. It is very frustrating to wait until the software has been built to learn that certain features are not feasible given the time or cost constraints, limitations of the implementation tools, or some other factor. An experienced team will lead the client company through the trade-offs, considerations, and costs of each of the target features. If there are features, which are not feasible in the context of other operational constraints, the client company can know this before any implementation effort is begun.

The advantage of the Software Engineering firm is the numerous experiences in Full System Life Cycle. They have designed, built, deployed, and supported numerous systems in their career. Collectively, for the team, this can be a very substantial number. They know what works. They know what has problems. They have not “seen it all”, but they have seen a very substantial portion of it. That experience and the knowledge associated with it is the foundation for project issue resolution and trade-offs. There will be projects that require a strategic prototype in order to address a specific risk factor (or set of risk factors). The risk factor(s) will affect end product capabilities and project timing. But greater the experience of the team, the more these risk factors will tend to be intrinsic to the project, and not associated with the expertise (or lack thereof) of team members.

Knowledge of timeliness throughout by milestone definition. The structured approach identified earlier also provides a series of measurable accomplishments during the course of developing the product. The completion phases are the "mega milestones". Each phase has well defined subsections. Some are sequential; others can be pursued concurrently. The client company can see clearly where the project is “on the map” to completion at any given point in time. In response to your inquiry as to when the product will be available for use, you really don't need to accept that programmer quip, "not yet" or "soon, very soon". The structured approach provides a schedule for each of the various phases enroute to the end product.

The use of Project Management tools (like MS Project), Component Calendars, and Change Control documentation increases the transparency of the overall Software Development project enormously. This gives management the needed information regarding progress vs time and costs. This directly increases the manageability of the project as well. Organizations are far more likely to behave in their best interests effectively with accurate and timely knowledge regarding a project status. Answers to questions like “How much will this feature cost?”, “How long will it take?”, “Can we get back on schedule if we delay these 2 features?” can be defined for well-managed projects. Otherwise management gets guesswork that often proves to be grossly in error. This is a major cause for projects being scrubbed, and time and money wasted.

Tools available. The tools available to the effort of software development are ever increasing in number, power, and expense. The days of designing on paper, writing all the source code, compiling each module, manually linking and debugging are part of history. Today we have tools, which assist in automating each phase of the project including Requirements Management, Design & Modeling, Development/Programming, Debugging, Documentation, Verification, Testing, Training, Security, Maintenance, Protection, and Help Desk. We have already discussed component re-use. Here we cover tools that are used by team members in performing their jobs.

Today we have tools for Requirements Management (most notably CaliberRM, Rational RequisitPro, and DOORS) that provide operations to systematically collect, articulate, refine, and publish systems requirements (their details, relations to other requirements and business sectors, refinement history or evolution, priority, date, author or proponent among stake-holders). We have Design tools that almost universally operate on UML (Unified Modeling Language). These provide design components that clearly and concisely define the business models, user operations, organization, workflow, automations, and other structural and behavioral properties of the client domain

and the system solution to address it. Most notable among these are (GDPro, Rational Rose, Visual Modeler, ParadigmPlus). We work with GDPro as it integrates very well with CaliberRM, DOOR, and ER/Studio (a database modeling, construction, and management tool). GDPro also generates very high quality C++ and Java code unfettered by tags and reserved areas. There is also an abundance of structured testing tools, help authoring tools, systems documentation tools (we rely heavily on the design definition and its refinement from the round-trip engineering relationship between source code and design provided by GDPro).

The software implementation and verification processes have been greatly enhanced by tools directed at streamlining their work. Tools to assist in limited source code generation exits for many of the development areas. Additionally, many tools exit today for code verification, debugging, execution monitoring, and testing. All of these can be used by the software engineering firm in providing the development service without additional cost - they incur no royalty fee for each installation. So the client company can benefit from the investments in software development tools made by the software engineering firm - the costs are amortized over many contracts. While software engineering firms tend to wield 4GLs (Fourth Generation Languages) more effectively than in-house programmers, these usually carry a royalty fee and in many cases do not offer the performance required. These development tools have come a long way in performance, but many still fall short of requirements for many projects. Companies, which have software development as their core business, have the armory of tools to perform this service in the most efficient fashion possible. Using these companies enables your products to benefit from the use of these tools.

Most cost effective use of costly talent. The development team may be required to consult with the most senior engineers on various isolated matters. The development effort benefits from the unique experience and depth of various resident specialists, but the client firm incurs only a small fraction of the costs of maintaining these individuals on the staff. The benefit is huge. The cost is very little. The involvement of a Database Design Engineer in a project may amount to 5 days. Their use is a very small fraction of the total manpower required. The difference to the project and product quality is profound. This benefit alone, in certain types of projects, will make outsourcing the definitively superior option.

Definable costs for product enhancements. This is really just a side-effect of number 2 above (Experience of the team). It is worth noting, as it is a source of on-going frustration for many companies with in-house programming staff, to get realistic estimates of employee time and calendar time required to provide a desired enhancement. The experience of the software engineering firm provides this ability. Additionally, being a contract organization for the client company, the client has substantial leverage in the form of performance agreements, to get the job done per design, on time, and within budget. When done in-house the enhancements are done when time permits, work is often disrupted by higher priority matters (e.g., bug fixes), and frustrations mount.

Implementation tools selected to fit the needs of the project, not programmer skills. This is normally much less a concern than the others, but in the effort approaching a comprehensive coverage of this topic, it has to be recognized. We hear about it more frequently than one would expect. A product or project is being developed with a particular language, tool, or technology because it is familiar to the developer(s), not because it is particularly well suited to the product or project. In many instances it is an either/or. Yes, there are trade-offs, but in many instances it is better to wield the tool of experience, which is not optimally suited for the project. However, there are products/projects for which one tool or technology is particularly well suited and others may be extremely poorly suited. Sometimes this mismatch has disastrous results. We could name some examples, but this would be from experience where we were brought into a project to rescue it and some readers might experience agitation over seeing the matter in print. Generally, these have been situations where the database application performance or scalability was insufficient for the domain, database replication schemas were inappropriate, data sharing schemas were wrong or poorly coded, the development language required excessive and obscure code to provide the target functionality, security issues were poorly understood, and target implementation configurations were not properly addressed early on.

Configuration, Training, and Help Desk Services. Training is an essential ingredient of a successful implementation. It is often spurned by programmers (especially "heads down" programmers). Having the outsource firm provide the on-site training and on-going Help Desk service will ensure a successful and full use of the product. If the product is going to be used by many users, the user community has a fairly high turnover rate,

or the product has many features some of which are needed infrequently, then an effective "Help Desk" service may be needed as well.

Configurations for the client-server-network under which the application will perform must be understood, tested, and documented. Defining the range of configurations expected throughout the organization(s) running the software and how the software will co-operate in those environments is vitally important to the stability, and performance of the software. The outsource firm should be expected to have expertise in this arena as well.

These are the salient benefits to outsourcing your IS needs to a software or IS engineering firm with many projects to their credit. How to select the best outsourcing firm is really a topic in itself and is beyond the scope of this paper. There are other benefits to outsourcing, but they generally are not so important as to provide a notable basis for outsourcing as opposed to in-house development.

III. BENEFITS FROM AVOIDING IN-HOUSE PROGRAMMING STAFF

Reduced payroll and fixed expenses. Naturally enough, if you outsource your software development you won't need any in-house programmers (designers, test engineers, technical writers, …). This is staff that draws pay all year, every year. Outsourcing removes this expense. When the required products have been completed what are these people doing? If you increase and reduce programmer staff based on product requirement cycles, what quality of people are you inclined to retain? The payroll is one of the on-going costs. The FICA, health benefits, disability insurance, 401K, office space, furniture, computers, phones, development tools, training, and all the supplies and other expenses are incurred with the in-house approach. A full year of a $70,000 per year programmer quickly becomes a $95,000 to $110,000 per year cost. Amortizing all of these costs over multiple projects with multiple organizations is the effect of outsourcing.

Freed Hostage - No dependence on employee continuity for software support and maintenance. Many organizations with in-house programmers feel that if they loose the programmer, the use of the product will greatly diminish. They get this feeling, most frequently, from the constant attention the programmer gives to the product. This should be sounding alarms and you should run (not walk) toward an exit, not further into the building! The outsource company should deliver a fully verified, tested, and well-documented program. It will usually need some minor modifications after delivery, but should be the target product and well behaved within several weeks on the outside. The length of this post delivery modification period increases with decreasing quality of the job performed in the design phase. If the design phase was never completed correctly, this post delivery modification syndrome can go on for a very long time.

Additionally, the outsourcing company is the service provider and should happily provide additional services even years later. If they are good at what they do their employee retention will be high and they will probably have the same staff available. They may not be the people who do the work, but they will be available for consultation with the people assigned to the follow-on project. Additionally, as the outsourcing company is experienced with such matters, they will have archived ALL of the materials developed in the earlier project(s). They will be able to come up to speed very fast and without much time invested by the client company.

Only scheduled, tested, and approved releases to the user community. It is not uncommon for the software developed in-house to be released into the user community without a schedule and without full testing and verification. Releasing software in this fashion tends to be very disruptive to the user community. A fix for one problem may create another, or alter a feature, or the behavior of certain modules. For well-designed and highly modular software there may not be any side-effects to changes in isolated areas of the software. But these two features often give way to the "quickly encoded" feature of in-house development efforts. It is not to say that there are not some extremely qualified and capable in-house software development teams to be found in companies. They are simply out populated by their less celebrated species.

Overall reduced costs, and no ongoing costs associated with the use of the software. The cost for providing a specific capability through outsourcing is definite and discrete, compared to the cost of

in-house being continuous and indefinite. Even if the outsourcing company is contracted to provide Help Desk service and some required modifications, this will be much less expensive than maintaining in-house programmer staff throughout the use of the product.

Ability to evaluate cost/benefit of any proposed enhancements. When using an outsource firm, you will receive (or should require) a cost estimate for the proposed enhancement. Highly complex enhancements may require a fee to perform the work in developing the estimate, but an estimate can and should be obtained. You can weigh the cost against the added benefit of the feature, and pursue the feature accordingly.

Decisions focused on core business of the company and users, period. Outsourcing allows you to focus fully and without compromise on the core business and the needs of the users servicing that business. In-house programming efforts must often be accommodated by compromising some set of needs in the user group in the form of features, performance, user-friendliness, or time to deliver. If the outsource firm is not fulfilling those needs, you replace them. The source code, documentation, and associated files and records are transferred much like patient files.

Reduced time of staff consumed in development effort. It may seem like the software engineering firm is consuming a lot of staff time with their problem definition and system design meetings, then reviewing and confirming tedious design definition issues. After the completion of this phase, very little time is required of the client firm. Properly conducting a software development project requires a great deal of staff time from the client company. The interviews and Joint Development Sessions (JAD) in the Requirements specification through Design Articulation requires a great deal of time and is disruptive. Meetings and JAD sessions should be highly structured with well-defined agenda for each meeting or session. The outsource firm should be experienced in this and should work to conserve the staff time of the client company and use those resources as efficiently as possible.

However, in-house staff tends to pull on personnel time in an ad hoc fashion. They often work under the same roof, or just down the hall. If they have a question or problem they'll just pop into to the appropriate persons' office and resolve the issues as they arise. While this sounds easy enough, it adds up to a very large consumption of time and impacts other activities and responsibilities of the personnel involved. It also reduces dramatically the structured and systematically defined design. Ad hoc techniques result in ad hoc design - which rarely arrives at a stable design definition. The demands on staff time of in-house programming efforts tends to be on-going - indefinite and continuous, just like the project.

Reduced time required to deliver product. In most cases, outsourcing will result in the product being provided in less time than that of in-house efforts. There are several factors which enable the outsource company to be more timely. The first being the breadth of the team - "many hands make little work". The other factors will depend on the nature of the product and the history of the outsource firm. If they have performed similar projects in the past, then direct experience, re-engineering existing code bodies, using object libraries, and other tools will play a major role in reduced time to deliver. This is not always the case however. A project cannot advance if the time and effort is not allocated to it. Some outsource companies tend to bite off more than they can chew. Naturally, this will not assist in a rapid delivery. In-house efforts can be made to drop all else and focus exclusively on a particular project - or jobs will roll. A similar dynamic can be applied to outsource firms through the use of target date incentives and penalties.

IV. CONCLUSION

Outsourcing is an excellent fit if your organization is characterized by:

1) Main focus of business is providing goods and/or services distinct from the Software Development domain.
2) The procedures, policies, business rules, user workflow, and information requirements are fairly well defined or definable,
3) Management and personnel have a comprehensive and accurate understanding of their business dynamics,
4) A good understanding or view is held of what capability is required of their IS (Information System) in order to resolve or address their needs,
5) The fundamental requirements of their information system are not highly dynamic - while specific requirements may arise weekly or monthly, the fundamental or central requirements do not change very often (annually or bi-annually, at most).
6) The details of the operation to be embedded in their information system are not so highly sensitive as to be protected by a solid non-disclosure agreement.

Most companies, which maintain an in-house programming staff, do so because of very good reasons. They should take a hard look at their reasoning to ensure that they are not sacrificing the benefits of outsourcing for the luxury of suffering through the costs and demands of providing these services in-house. Below are the most common reasons management decides on providing in-house staff to accommodate the company's IS needs. These may or may not be true. Careful consideration of the company's actual situation is required to come out ahead with whatever strategy they employ.

1) The company IS requirements are very dynamic, and outsourced services would be too slow in providing their needed capabilities. Many organizations who see their IS needs as being too dynamic to be compatible with the outsourcing strategy, actually have a fairly static set of IS needs, but a highly variable understanding of those needs. Additionally, the rate at which the priorities among their various IS requirements change, should not be confused with the actual set of fundamental requirements of their IS. If a company can list and agree on the high level, information oriented requirements of their IS at any given point in time, and if each subsequent list contains additions of lesser or equal importance to the organization than the entries on each previous list, and they are small in number relative to the preceding lists, then their needs are much less dynamic than those which warrant an in-house staff.
2) They need a high degree of control over the IS in order to assure that it meets their needs,
3) Their IS needs are sufficiently unique to require and warrant an in-house staff,
4) They can provide for their IS needs more effectively than could a third party.

Outsourcing is a great means of meeting your software capability needs. It may or may not be the best means of serving those needs. Hopefully, this article will assist you in weighing the various factors which will determine which approach to employ.