"IT Litigation & the Bad Contracts that Foster Failure: *& What YOU Can Do About It!"*

A COMPANION THOUGHT PIECE FOR THE 3.5 HR. MCLE PROGRAM OF SAME NAME

This report is NOT a standalone article. *It is a companion thought piece to be used in conjunction with the digitized recording and accompanying detailed slides at: www.attorneycredits.com for the "IT Litigation & the Bad Contracts that Foster Failure – & What YOU Can Do About It!" MCLE presentation.*

It provides some additional context and knowledge about the successful IT contracting model to help the listener/learner do a better job in developing, negotiating, documenting and managing IT contracts. Some additional areas, not in the presentation are included here that I believe may be helpful to attorneys wishing to improve their knowledge and skills in IT contracts, IT project turnarounds, IT deal making and IT litigation.

My thoughts are organized into four sections:

I. <u>Why Systems Fail?: Background, Statistics and Experiences</u>

- A. IT Litigation Claims
- B. Statistics on Successful, Challenged and Failed Systems
- C. System Failure Examples
- D. Why Systems Fail?

II. <u>The Role of IT Contracts</u>

- A. Why Are Contracts So Important?
- B. The Makeup of Your IT Contracting Team
- C. Most Lawyers Leave Out the Critical Contract Clauses in their Contracts
  1. The Lawyer Side of the Contract (What most lawyers do best!)
  2. The IT Management, Project Manager, SME Technical Expert Side

III. <u>Method for Achieving IT Contract Success</u>

- A. Defining Success
- B. The Systems Approach to IT Contracting
  - The contract must relate to the SDLC phases and tasks
  - The contract must also cover/relate to the changed corporate environments
- C. Eliciting and Defining Requirements – Why Still So Difficult?
  - i. So Many Stakeholders – and Many with Different Needs
  - ii. Sources of Requirements
  - iii. Types of Requirements
- D. Changing GUI (Graphical User Interface) Requirements
  1. DD: The Digital Dinosaur (~70 years and up)
  2. DI: The Digital Immigrant (~40-65 years old)
  3. DN: The Digital Native (approximately 20-35 years old).
  4. DNA: The Digital Native Always (approximately 5-15 years old)
- E. Risks Known BEFORE Project Starts!

Preface

In my experience at WSR Consulting Group, LLC covering nearly 100 litigation matters involving software and project failure in the US, Canada, Europe and Asia and in many different courts, it has become clear that the same themes, allegations, claims, responses, counterclaims and defenses appear time and time again. Depending on the specific facts in the instant situation, some arguments are persuasive, others are red herrings.

In the Successful IT Contracts Model (at http://www.wsrcg.com/PDFs/model_itcontracting.pdf), I have combined Attorney Richard Bernacchi's "Systems Approach to IT Contracting"[1] from his seminal works with my expertise in developing software/systems, and lessons learned as an expert witness opining on system/software project failure disputes to provide an organization, structure and approach to developing pertinent, and perhaps "best", IT contract clauses and practices. The approach, understandings reached between the parties during the contract negotiations, sand the resulting contract can be critical in helping **prevent/mitigate** IT project failures, **AND** winning a dispute in court -- should it come to that.

I. WHY SYSTEMS FAIL?

A. IT Litigation Claims

In virtually all large-scale systems project failures where I have been asked to be an expert witness during the last 21 years (in the US, Canada, the Caribbean, Asia and Europe, and in the U.S. Court of Federal Claims), the dispute between the parties always seems to boil down to some combination of same complaints (see Table 1 below), with the customer complaints reflected in the **"WE SAID…"** column, and the vendor/developer, integrator, implementing VAR complaints reflected in the **"THEY SAID…"** column.

See Table 1 below:

[1] First brought to my attention by Attorney Richard L. Bernacchi, an extraordinary IT Lawyer, and taken from his truly seminal work, "Bernacchi on Computer Law: A Guide to the Legal & Management Aspects of Computer Technology". Published and updated by Little, Brown & Company from 1986 through the late 1990s. Mr. Bernacchi, senior partner at Irell & Manella LLP, specializes in the legal, technical and strategic planning issues arising from computer hardware and software, multimedia, Internet, telecommunications and other advanced technologies, including development, acquisition, protection and exploitation of IP; acquisitions and mergers; licensing, distribution, Web site development, co-marketing and joint ventures.

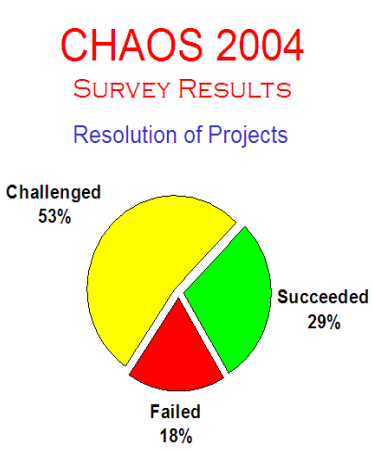
	"WE SAID…" *CUSTOMERS/USERS*	"THEY SAID…" *VENDOR/DEVELOPER, INTEGRATOR, IMPLEMENTER*
1	**Limited Functionality** Several critical requirements we discussed & that you agreed to were NOT delivered. System has little/no value without them	**Customer Kept Changing Scope** Every time we got ready to test you changed the requirements and scope of the system. You refused to follow the indentified change control process.
2	**System Doesn't Work** What do you mean it doesn't provide for international currencies? Wasn't that implied? We discussed that!	**Customer Changed Mind** You had a merger & no longer want or need the system
3	**We Can't Use It** No one knows what to do with the results of the system. We were not properly trained.	**Client People Not Trained** Your best & promised attendees got called back to your office for emergencies
4	**System Failed In Field** No one knows what to do w/ system results. Our old (current) org structure does not support system needs	**Client Did Not Do BPR** You were responsible your non computerized processes so they would work with the new systems – and you didn't!
5	**Fundamental Flaws** System is fundamentally flawed. A standard PC-based POS system will not work in a fast food restaurant w/ flour & oil contaminants in the air.	**Only "Two More Months"** Even though you approved two more month delays several times, give it to us one more time and we'll fix everything!
6	**They Should Have Told Us** You never told us that we would need dramatically more hardware & additional staff to manage, operate, maintain, backup & secure the system	**They Wouldn't Listen** We told you on several occasions including in writing. However, you were not willing to reserve the money until you absolutely needed to.
7	**System Is Full Of Bugs** Most of the reports have errors. Virtually none of the inquiry responses are correct.	**Bad Data/Conversion** You didn't cleanse old data from errors before sending to us to map/create new database. Garbage In – Garbage Out!
8	**Developer Failed @ SIPM** You were the Systems Integration Project Manager (SIPM). It was your job to ensure the hardware, software, netware, site preparation, implementation and training ALL WORKED TOGETHER	**Customer Failed @ SIPM** You were the Systems Integration Project Manager. It was your job to ensure the hardware, software, netware, site preparation, implementation and training ALL WORKED TOGETHER
9	**Poor Advice** You never told us that we had to decide on XYZ by November 10th to keep the project on schedule	**Poor Customer Decision-Making** We told you the schedule impact of your late/shifting decisions. Your tardiness made us miss critical path dates.
10	**Unqualified Personnel** You told us that we would get your very qualified software pros with expertise in your SDLC method, implementation approach & testing regimen… like the staff that presented the demos. We got none of that!	**Wrong Client People** You promised to assign your Subject Matter Experts to help us define requirements, develop test scenarios, and be "trained as trainers." They were not made available to participate when needed & promised. We got none of that!
11	**Wrong Development Process** You promised to follow your SDLC in modifying & implementing system. But you cut steps & corners which ultimately caused us to redo/retest large parts of the system & miss our cost and scheduled targets	**Poor Client Support** You would not let us follow our promised methodology because you thought our work was good enough and you did not want to pay for full and proper testing. We told you the risks and you decided to push ahead knowing them.

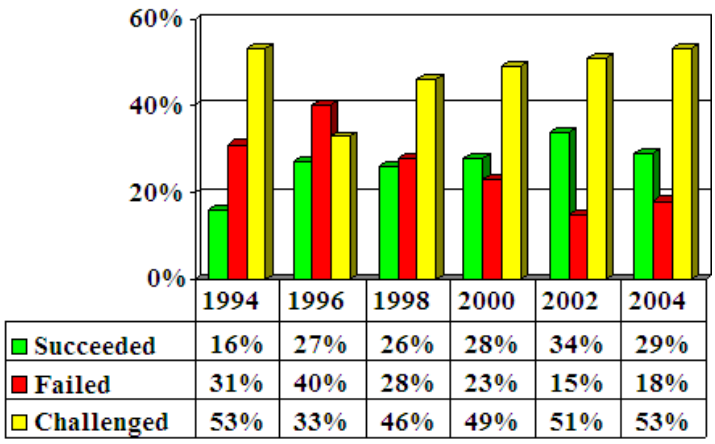
Whenever I present my "We Said – They Said" to audiences, litigants and attorneys, it is met with guffaws, embarrassment & heads shaking yes. **If you too are smiling as you read this, then this program is for you!**

B. Statistics on Successful, Challenged and Failed Systems

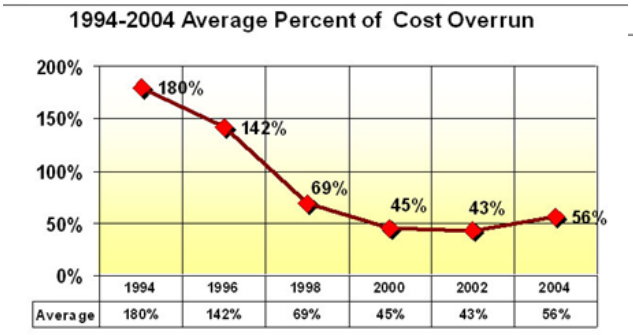
With all of the "panaceas" introduced over the decades to help ensure successful system solutions, such as structured methodologies, JAD, object-oriented programming, test coverage analyzers, COCOMO models, function point metrics, CMMi standards, automated tools, agile methods, TQM, inspections, static testing and walkthroughs, and more, successful systems and IT projects are still elusive.

In fact, statistics show little improvement over the past 15 years regarding the % of systems project successes, and the number of challenged and failed/scrapped systems. The Standish Group, through its "Chaos Reports," publishes the following rates of success and failure, and cost and schedule overruns on systems projects. While it is possible to find some flaws in their methods to obtain and calculate project statistics, the Standish Group's statistics are not materially inconsistent with other studies that report such statistics reported periodically[2].

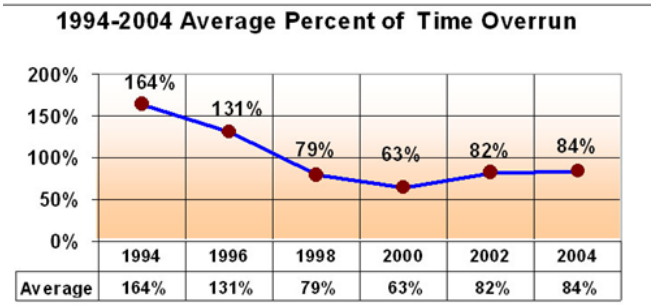




	1994	1996	1998	2000	2002	2004
Succeeded	16%	27%	26%	28%	34%	29%
Failed	31%	40%	28%	23%	15%	18%
Challenged	53%	33%	46%	49%	51%	53%



	1994	1996	1998	2000	2002	2004
Average	180%	142%	69%	45%	43%	56%



	1994	1996	1998	2000	2002	2004
Average	164%	131%	79%	63%	82%	84%

In my opinion, a success statistic of 34% is only good for baseball sluggers with a .340 year.

UPDATE – CHAOS Summary 2009:
- **+ success = 34% (delivered on time, budget, on target)**
- **+ challenged = 44% (late, over budget, < required F&F)**
- **+ failed = 24% (cancelled before complete; delivered & never used)**

"… numbers represent downtick in success rates from previous study, and significant increase in number of failures … low point in last five study periods & highest failure rates on over a decade.

[2] See http://www.it-cortex.com/Stat_Failure_Rate.htm#The%20KPMG%20Canada%20Survey%20%281997%29
Robbins-Gioia Survey (2001); Conference Board Survey (2001); KPMG Canada Survey (1997); OASIG Survey (1995)

C. System Failure Examples

Below are some well-known examples of systems and project failures over the past several years.

YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of $100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after $170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after $33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to $445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to $100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after $130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing $25 million.
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to $151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of $50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after $4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after $40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to $3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause $350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after $25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after $2.6 billion is spent.
1994	State of California	DMV system canceled after $44 million is spent.
1994	Chemical Bank	Software error causes a total of $15 million to be deducted from 100 000 customer accounts.
1993	London Stock Exchange [UK]	Taurus stock settlement system canceled after $600 million** is spent.
1993	Allstate Insurance Co.	Office automation system abandoned after deployment, costing $130 million.
1993	London Ambulance Service [UK]	Dispatch system canceled in 1990 at $11.25 million**; second attempt abandoned after deployment, costing $15 million.**
1993	Greyhound Lines Inc.	Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of $61 million.
1992	Budget Rent-A-Car, Hilton Hotels, Marriott International, and AMR [American Airlines]	Travel reservation system canceled after $165 million is spent.

Sources: Business Week, CEO Magazine, Computerworld, InfoWeek, Fortune, The New York Times, Time, and The Wall Street Journal Compiled by Why Software Fails author By Robert N. Charette

D. Why Systems Fail?

The complaints summarized in the **"We Said … They Said …"** model (Table 1), derive from key underlying factors and root causes of systems failure that emerge time and again the following areas of **systems development task and project management process**:

- Unrealistic system or project goals (based on company resources, culture, staff skill levels, and novelty of the system).
- Unrealistic/unachievable timetables
- Poor estimates (cost and schedules) regarding project resources and staff needed
- Incorrect, non-rigorous, or implied:
  - functional requirements
  - performance requirements
  - testing requirements
  - data conversion process and requirements
  - training regimen
  - maintenance requirements
- Confusing, poor and "rosier than actual" reporting of the project status
- Unspecified roles and responsibilities of the parties
- Unidentified, unallocated and unmanaged risks
- Differing expectations and poor communications between customers, developers, integrators, and users
- Failure to understand and manage IT project complexities, logistics and stakeholder expectations
- Failure to follow good Software Development Life Cycle (SDLC) practices[3]
- Ignoring good project management processes[4]
- Conflicts and clashes between and within Acquirers, Vendors, Users, and Internal MIS stakeholders[5]
- Use of immature or bleeding edge tools and technologies; failing to use mature tools and technologies
- External and internal commercial and political pressures

These underlying factors and root causes can be addressed and mitigated by writing better IT contracts.

---

[3] Systems Development Life Cycle (SDLC) is any logical process used by systems project managers and analysts to develop an information system, including requirements, development, testing, training, systems implementation and cutover, and maintenance and user ownership. Any good SDLC should result in a high quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works effectively and efficiently in the current and planned Information Technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

As computer systems have become more complex, new SDLC models have been created: "waterfall," "spiral," "rapid prototyping," "incremental" along a spectrum of agile → iterative → sequential. Agile methods, like XP & Scrum, focus on light-weight processes allowing rapid changes along the development cycle. Iterative methods, like Rational Unified Process and Dynamic Systems Development, focus on limited project scopes & expanding/improving products by multiple iterations. Sequential models, like Waterfall, focus on complete/correct planning to guide large projects and risks to successful, predictable results.

In project management a project has both a life cycle and a "systems development life cycle," during which a number of typical activities occur. The project life cycle (PLC) encompasses all activities of the project, while the SDLC focuses on realizing product requirements. Source: http://en.wikipedia.org/wiki/Systems_Development_Life_Cycle

[4] Notably, failure to implement and/or use appropriate: Earned Value Analyses; Estimates to Complete; Critical Path Methods; timely Variance Analyses; assigned and monitored Issues Logs; problem escalation processes; Steering Committee; scope management procedures; and business change management processes (Business Process Re-engineering – BPR)

[5] See Warren Reid's "Riskapedia Model" at **http://www.wsrcg.com/PDFs/model_riskipedia.pdf**

SECTION II. THE ROLE OF IT CONTRACTS

A. Why Are Contracts So Important?

The essential role of an IT contract, is to **allocate project and system risks**. However, a poorly written contract has the effect of increasing the overall risks for both parties. A well-written contract has the power to take risks off the table so that neither party has to contend with it or that one party agrees to take on that risk in return for something.

Because the different constituent parties have innately different goals, the contracting process is the process to bring the differences out into the open so they can be discussed and agreement as to who will pay for and absorb the risks can be reached (or traded off). Poorly written contract provisions, poor precision in the contracting language and the omission of certain contracting provisions creates and increases risks for both parties.

It is generally true that the Standard Contract form" favors the large contractors," if it was drafted by them. However the greater sin in using them is that as contracts meant to be applied in many situations, standard form contracts most often fail to address the peculiar details of any one situation. The standard language gives a sense of false comfort, and lulls one or both parties into believing that the details of the arrangement have been properly addressed, But incomplete and ambiguous provisions will haunt the parties and the project itself – because once the drafters are gone, no one will remember what was meant, and the contract will be powerless to guide the parties and project through the tough questions that invariably arise..

In my opinion and experience, **a good IT contract process that fosters project success:**

- ✓ Provides clear & explicit delineation of each party's rights, obligations & expectations, roles, responsibilities and deliverables
- ✓ Should be the reflection of the identification, negotiation and acceptance of the other party's views, needs, beliefs & objectives **BEFORE** a deal is made. (We have seen many contracts that should never been signed as it should have been clear that one party could not possibly absorb the risks, schedules and requirements of the contract nor meet its roles and responsibilities.)
- ✓ Generally produces a much better:
  - o working relationship during future calm & emergency moments that invariably arise during the project
  - o increases the prospect of project success (i.e., delivering on budget, schedule, target)
- ✓ Produces a living document – i.e., a contracting team and a living contract that will be referred to and be adaptable to changes

B. The Makeup of Your IT Contracting Team

In my experience, **a solid contract will only emerge from the involvement of representatives that are empowered by their respective parties and actively engaged in the creation and management of the project. Contracting teams should include the following empowered members**:

- ✓ A **CXO** who can/will make decisions regarding the allocation and acceptance of risks
- ✓ A **CFO** who will evaluate the alignment of the contract clauses to the business case and accept financial responsibility
- ✓ The **project champion, subject matter experts, project managers and technical staff** (as needed) who together have a strong sense of **SYSTEMS VALUE** (to establish requisites for required systems and project quality, and to determine project/system standards & acceptance criteria, and future service level objectives)
- ✓ A **lawyer** with appropriate legal skills and experience

This team should be able to address matters of fact, law, evidence, risk and technology if and when needed in the present and in the future.

C. Most Lawyers Leave Out the Critical Contract Clauses in their Contracts

In the overwhelming majority the contracts that I see in court and at the center of alleged IT systems and project failure, there is a total lack of appreciation for and content regarding the Systems Development Like Cycle (SDLC), its tasks, its processes and its interim deliverables with a description of the quality/acceptance criteria for each deliverable.

Too many lawyers and contracts focus only on the things that each lawyer knows best – i.e., the purely legal clauses in the contract, many of which are important and necessary, but even when taken together, **are clearly insufficient regarding: (a) resolving project disputes during the project life cycle, (b) turning around "runaway" projects, and/or (c) supporting a winning litigation strategy and claim.**

1. THE LAWYER SIDE OF THE CONTRACT (WHAT MOST LAWYERS DO BEST!)

- ✓ Basics
- ✓ Set Stage
  - o Recitals
- ✓ General Provisions
  - o Parties To The Contract
  - o General Reps & Warranties
  - o Definition Of Terms
  - o Assignment Of Delegation
  - o Interpretation Of Agreement
- ✓ Ownership & Protection
  - o Title
  - o License Rights
  - o Proprietary Rights Indemnity
  - o Confidentiality & Security
- ✓ Risks & Rights
  - o Risk Of Loss/Damage
  - o Insurance
  - o Price Protection
  - o Renewal Options
  - o Purchase Options
  - o Trade-In Rights
- ✓ Termination & ADR
  - o Term & Termination
  - o Limits & Exclusions Of Liability
  - o Taxes
  - o Miscellaneous Protections
- ✓ Dispute Resolution Mechanisms

In addition to the lawyer areas note above, there is another absolutely critical part of the IT contract which is all too often missed, ambiguous or very poorly done. This part follows the path of the SDLC and identifies: how the project is to be managed; the roles and responsibilities of each party; the identification and sequencing of life cycle tasks and the estimate of costs, schedule and hours required to complete them; problem and issue escalation procedures; the scope of the systems and the project; the costs, schedules, functional, quality and performance requirements and "-abilities if the new system; the proper resources required; the inspection and approval of interim deliverables; and much more.

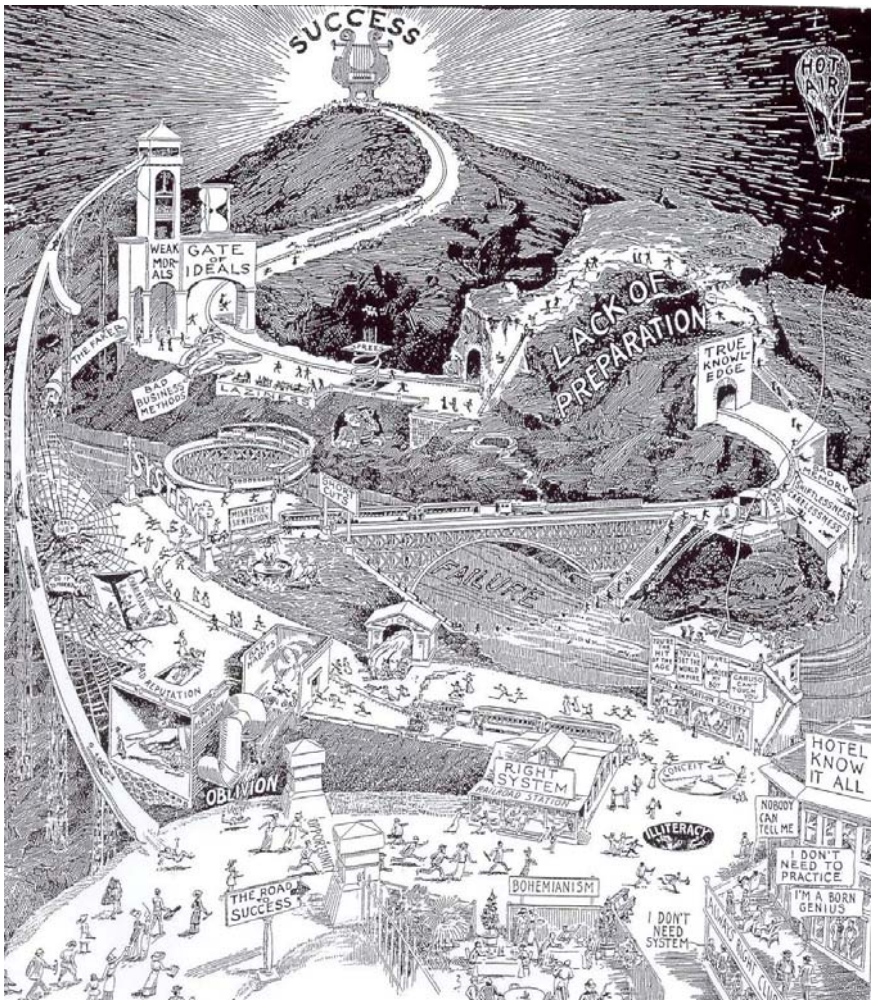
Shying away from these tough clauses only hurts the project and its chances for success. **These contract clauses cannot be prepared by the attorney alone – no matter how smart he or she is.** They require an **IT Contracting Team** consisting of financial, technology and management staff intimately familiar with the developed/licensed system's history, intended goals, definition of success, the required specific system tests and ongoing measurements to be met to provide ongoing system relevance (upgrades) value to the company, among many other things.

2. THE IT MANAGEMENT, PROJECT MANAGER, SME TECHNICAL EXPERT SIDE

- ✓ Systems Description
- ✓ Functional Requirements & Perform Measure
- ✓ Project Management
  - o Project Timetable
  - o Project Management & Reporting
  - o Project Costs & Schedule Payment
- ✓ Pre-Go Live Items
  - o Site Preparation
  - o System Configuration & Installation
  - o Training
  - o Documentation
  - o Low Level, System & Integration
  - o Acceptance Testing
- ✓ Customize & Convert
  - o Custom Programming Requirements
  - o Conversion & Support Services
- ✓ Personnel
- ✓ Maintenance
- ✓ Warranties
- ✓ Special Outsourcing Considerations

SECTION III: METHOD FOR ACHIEVING IT CONTRACT SUCCESS

A. Defining Success



A good IT contract and the IT contract process creates a mutual understanding between the parties of what is the definition of success for the project and what each party must do to reach that goal.

Different constituents likely have different goals for the project. For instance:

- **Users may opt for:** many features, changing requirements, applications compatibility, high levels of service, flexible contract and early availability of the system
- **Acquirers may opt for:** mission cost effectiveness, limited development budget and schedule, compliance with regulatory standards, cultural sensitivity, visibility and control over development, a rigorous contract, system with high privacy and security components
- **Vendors, Developers, Integrators or Consultants may opt for:** a flexible contract, ease of meeting budgets and schedules, stable requirements, freedom to chose and execute the SDLC they desire, freedom to choose the members of the project team(s), freedom to reuse code it already has or to select commercial of the shelf design/coding components

Even within departments, such as MIS, the Development Group may have different goals and definitions of success than the Maintenance Group regarding: what should be provided through in-house development vs. outsourced vs. bought as COTS (Commercial off the shelf, e.g., SAP e-business suite, Oracle's manufacturing suite, PeopleSoft's Integrated Human Resources products), ease of transition and acceptance criteria to move system from development to maintenance, ease of maintenance, total cost of ownership, voices in the acquisition of new systems, software, hardware and netware.

Often the differences in objectives and the competition between goals and definitions of success among stakeholders, have caused projects and systems to be scrapped, or at least heavily challenged.

So as you begin to draft up your outline for the IT Contract, ask yourself, and others, just what is the definition of success in this instance? How will you measure it? Is a "successful" project/system one that meets any/most/all of the following measureable goals?

- Stays on budget? Meets the planned or drop-dead schedule(s)?
- Delivers all of the critical requirements?

- Is well designed and documented that enables easier and less risky maintenance and security?
- Has the ability to interface with new technologies on the horizon?

- The Mean Time Between System Failure (MTBF)?
- Being the first to be able to get your products to market?
- The lowest Total Cost of Ownership (TCO) of the system and project over its life cycle?
- Allows a company to get its next round of financing?
- Meets the stated business case(s)? Meets Return on Investment (ROI) requirements? Enables corporate sales growth? Reduces costs? Increases gross margin? And/or reduces working capital requirements?
- Has the ability to grow sales with less staffing?
- Enables better customer service?
- Allows the company to better compete in an ever more complex marketplace?

B. The Systems Approach to IT Contracting[6]

The contract must relate to the SDLC phases and tasks

It is my belief and experience that the best way today to develop a successful and inclusive IT Contract, and a real appreciation and meeting of the minds between the parties, is to:

1. Relate the contract clauses to the individual phases and tasks in the agreed to SDLC. Identify responsibilities, interim deliverables, acceptance processes, escalation regimen, scope change and approval processes. If you are contracting for results and NOT resources, your contract must reflect that. Penalties for contract infractions should be spelled out and provide incentive for the failing party to continue to perform – but in a better fashion. Critical path must be established and understood by the parties. "Earned value" should be used as necessary. Deviations from agreed to methodology must be approved in advance. The contract should require S*M*A*R*T[7] measurements to assure that the system is meeting requirements.
2. Tie contract requirements to the promises made in the Statement of Work documents
3. Define and enforce agreed to project management, reporting and escalation practices
4. Keep in mind that:
    a. "Any procurement of data processing products or services involves a *system.*
    b. A system consists of an integrated combination of equipment, software, related services, and buyer and seller (licensor) involvement necessary to achieve the specific business results desired by the buyer within a specific timeframe and within a specified budget.

[6] First brought to my attention by Attorney Richard L. Bernacchi and taken from his truly seminal work, "Bernacchi on Computer Law: A Guide to the Legal & Management Aspects of Computer Technology". Published and updated by Little, Brown & Company from 1986 through the late 1990s.

[7] **S*M*A*R*T:** Specific, Measurable, Attainable, Relevant & Trackable metrics and measurements (e.g., new function points developed; number of errors discovered and fixes requested; time to repair bugs tracked over time; time to respond to change requests; time to implement change requests; criticality of defects found over time; mean time to defect (MTTD); computer resource availability and utilization over time; response times during normal and high loads; downtime over time; number of transactions that can be processed during overnight window; security breaches; database lookup times; maximum number and types of errors for system to be accepted; mandatory statement of quality and frequency of training and training documentation and courses for staff and new systems users, etc.

c. Not all of the elements or ingredients of a system are necessarily involved in each procurement.
d. Not all of the products and services will necessarily be provided by a single vendor.
e. Some of the services may be provided by the buyer.
f. Not all products and services will necessarily be implemented to put into operation at the same time."

The scope of these considerations must cover distinctively different time periods and SDLC project phases that are found in **virtually all "good" SDLC methodologies:**

- ✓ Pre-project business case reflecting business goals & expected costs, benefits, ROI, TCO, etc.
- ✓ Define and prioritize high level requirements and user profiles
- ✓ Define project scope, schedule and budget and risks
- ✓ Prepare preliminary functional and technical design
- ✓ Develop business process recommendations
- ✓ Prepare detailed functional and technical design
- ✓ Develop project plan. Revise budget, schedule, staffing requirements, and risks
- ✓ Develop the software to meet both the functional and performance requirements
- ✓ Develop and execute a strong testing & acceptance regimen with appropriate interim signoffs
- ✓ Establish and deliver training and documentation
- ✓ Cleanse, convert and test data conversion
- ✓ Establish help desk and post-implementation debugging processes, teams and logs
- ✓ Assure pre Go-Live readiness
- ✓ Deploy business processes
- ✓ Roll out new systems/software
- ✓ Manage the post-implementation settling period
- ✓ Turnover systems to maintenance function

Each of these systems phases/tasks, should have corresponding contract clauses which identify specifications for completion of the tasks, each party's responsibility, the measurements/metrics defining the acceptance criteria for each major task and submitted deliverable.

Today, many IT projects include multiple systems linked/integrated together, the use of Internet for a myriad of purposes, the use of mobile computing, wikis, and perhaps web 2.0 applications. In addition, the quality of the delivered product(s) and its maintainability have become ever more critical – as the importance of good, timely, reliable information needed for improved decision making has become more critical to buyers of systems.

The contract must also cover/relate to the changed corporate and human resources environments needed to support the new system capabilities

Mr. Bernacchi states: "The systems approach emphasizes the fact that a customer is NOT just procuring a piece of hardware or a series of programs, but is changing in fundamental ways the company's business processes or one of its units. Thus, in the systems approach, the interrelationships between, on the one hand, the hardware, software and services provided by the vendor, and, on the other hand, the buyer's organization -- its people, its culture, its modes or doing business and its financial constraints -- **all have to be considered.**"

Accordingly and importantly, the **contract must contemplate and address not just the "computer" aspects of a system**, or the development aspects of a project, but also address the non-computer aspects of a system, and the non-development aspects of a project, including: the company's new work flows; altered decision patterns and decision-making processes (now that the new system will provide analytical and predictive information; changed organization structure and reporting relationships; shifting job responsibilities; BPR; new corporate policies; post go-live change management, maintenance, services, production support, etc.

C. Eliciting and Defining Requirements – Why Still So Difficult?

It is extremely critical that the project teams and the contract get the requirements right. Why? …Because wrong, incomplete, or ambiguous requirements will result in a faulty design.

This faulty design will cause the generation of bad code (i.e., code that may meet the requirements of the faulty design – but not meet the users' requirements and expectations). The bad programs will be tested against false test criteria and inappropriate tests based on the faulty designs. The team will then document and train for an unacceptable system that "WORKS AS DESIGNED!" But certainly doesn't work as expected or intended!

Fixing such systems essentially requires the party to almost lose all of the already performed and produced – requiring a "do-over!"

i. So Many Stakeholders – and Many with Different Needs

Good requirements are hard to come by because there are so many stakeholders that must be addressed and many have different information and control needs and views on what the successful system must do, and how it must operate and perform. Stakeholders include:

- ✓ **Users & Support Groups**
  - o Partners
  - o Persons most knowledgeable
  - o Subject matter experts (SMEs)
  - o Various classes of users
  - o Department heads/representatives
  - o Systems and software developers
  - o Systems and software integrators
  - o Industry consultants; Project consultants
  - o Project Steering Committee
  - o Acquirers (i.e., the company's top executives)
  - o Regulators
  - o Legal
  - o Human Resources and Human Factors specialists
  - o Inside and outside auditors
  - o Operations group
  - o Maintenance group
  - o Documentation specialists
  - o Help desk
  - o Database administrators

- ✓ **Implementers**
  - o Program manager
  - o Project manager
  - o Test manager
  - o Project support staff
  - o Trainers
  - o Systems and software architects
  - o Out-Sourcers
  - o Project champions and sponsors
  - o GUI interface specialists
  - o Business Analysts to perform Business Process Reengineering (BPR) before system is implemented

In some cases, you'll also have to meet the requirements (and sometimes, "fancy") of members of the Board of Directors ... some of whom really understand IT and the capabilities, risks and challenges of building, integrating and implementing successful systems, and some who may get little appreciation of IT and/to get their ideas from their MIT genius niece or otherwise crazy brother-in-law.

ii. Sources of Requirements

It is not enough nor appropriate to simply replicate in a new system the functionality and procedures used in current/old systems and then graft onto that the improvements that users would like to see. Some users may only know the current failing system they are working on, and that way of doing things – and are unable to think/imagine beyond their current systems' problems de jure. That is why I recommend that the following sources be used in creating a good requirements list.

- ✓ Users at all levels of the organization including powers users and the preparers of the company's Strategic Business Plan – who know where they would like to bring the company.
- ✓ The people who developed the Long Range Systems plan, and then adjusted it to be aligned with the company's Strategic Business Plan

- ✓ Industry experts – who know where your industry is headed, and which systems will help take the leading companies in the industry there.
- ✓ IT Consultants who specialize in systems solutions in your industry
- ✓ Vendor User Groups
- ✓ Your competitors
- ✓ Internet and vendor sites showing each vendor's systems capabilities and comparisons to their competitors
- ✓ And industry technology disruptors (i.e., upcoming technological breakthroughs that may cause you to rethink your requirements and new system capabilities, e.g., web 2.0 and social networking, cloud computing, mobile computing, smart phones, etc.)

iii. Types of Requirements

In addition, requirements come in many flavors: feature and functional requirements; performance requirements; and usability requirements to name a few. There are many levels of requirements needed from different including:

1. **Business requirements** including:

1.1. – Commerce, trade, industry, selling & production requirements
1.2. – User, Management Information Systems requirements
1.3. – Software, Operations, Maintenance requirements
1.4. – Functional, Business Process Reengineering, Outside constraints

2. **Quality requirements** including:

2.1. – Developer, User, and Performance quality requirements
2.2. – Software hardware, netware, and platform/architecture

3. "-ability" requirements including:

3.1. **For (external) Users:**

3.1.1.	Availability	Dependability	Flexibility
3.1.2.	Integratability	Modifiability	Operability
3.1.3.	Reliability	Robustness	Securability/security
3.1.4.	Usability	Scalability	Safety
3.1.5.	Survivability	Restart & recoverability	Configurability

3.2. **For (internal) Developers & Techs:**

3.2.1.	Adaptability	Auditability	Deployability
3.2.2.	Extensibility	Interoperability	Maintainability
3.2.3.	Portability	Performance abilities	Reusability
3.2.4.	Testability	Manufacturability	Traceability

Furthermore, since it is virtually impossible to maximize each of these requirements for a system of any complexity within a reasonable time and for a reasonable cost, the stakeholders, under project management, consultant and Steering Committee leadership, will have to prioritize the requirements, and perhaps drop, defer, simplify, clarify, trade-off and otherwise resolve conflicts between requirements.

Requirements must be derived not only for the current operations and climate, but also for the most likely business opportunities and scenarios in the near future.

D. Changing GUI (Graphical User Interface) Requirements

Another issue worth mentioning is the requirements for the GUI[8] interface.

As the recession perpetuates older workers staying in their jobs and in the work force longer (a phenomenon that actually started even before the recession). Due to this trend the GUI for a single large system (such as an ERP system) now has to meet the needs of many categories of users in a single large workplace. The categories below have been around for some time, except the DD and DNA category which I created. They are not perfect and surely many people do not easily fit into the category most associated with their particular ages – but the theory is correct and imposes additional challenges and responsibilities on the requirements team and the design/coding/user acceptance teams.

Note that I have left 5 years between each category because the model is not that precise of a production tool.

1. **DD: The Digital Dinosaur (~70 years and up)**
   This group is working longer into their later years. Learning new systems can be scary and difficult as the willingness to change becomes weaker. Their speed on the computer tends to become slower. Their GUI is quite complicated, as it is not only a technical issue but also an age related issue regarding how our brain works.

2. **DI: The Digital Immigrant (~40-65 years old)**
   The DI, like myself have some knowledge and comfort with technology – but really didn't grow up with the type of advanced computing and computers available to day. This group has the most variation in computing talent and requires a more step-by-step, help oriented GUI as compared to the Digital Natives. If they don't use an application, module, or service often, they will tend to forget how to access it and properly use it for their needs. Multitasking becomes more confusing over time. A slower, more formulaic, step-by-step approach to learning and using new systems and technology is required.

3. **DN: The Digital Native (approximately 20-35 years old).**
   The DNs were mostly in their teens when the World Wide Web came into being. They studied and used PCs in school and growing up. They embraced the Internet, texting, instant messaging, personalized ring tones, Google, iTunes, Facebook, twitter, smart phones, Skype and apple computers before and more than any other age group. They are truly and literally wired for computing. They understand how games work without ever reading the instructions — they just, for the most part, intuitively know! At work, they will want systems that get them to where they need to be in 1 or 2 clicks. They want to get in and out – while multitasking.

4. **DNA: The Digital Native Always (approximately 5-15 years old)**
   You can see them at the Apple Store. Five-year old girls and boys, bouncing on large air-filled exercise balls as they use a keyboard with dexterity to play various computer games and out wit computer friends and foes.

[8] (Graphical User Interface) A graphics-based user interface that incorporates movable windows, icons and a mouse. The ability to resize application windows and change style and size of fonts are the significant advantages of a GUI vs. a character-based interface. GUIs have become the standard way users interact with a computer, and the major GUIs are the Windows and Mac interfaces

E. Risks Known BEFORE Project Starts!

Lessons from past experiences with IT contracts, and the risks that arise from poorly drafted contracts can be summarized in a checklist of risks that should be checked-off as the drafting process proceeds. **You now know many of the risks and reasons for failure that plague virtually every IT project** – and because you now know these risks beforehand, you can better plan, contract for, mitigate, monitor and manage them!

KEEP THE FOLLOWING RISK AREAS IN MIND BEFORE YOUR CLIENT BEGINS THE PROCESS OF SYSTEMS SELECTION AND/OR NEW SYSTEM DEVELOPMENT AND BE PREPARED TO ADDRESS THESE IN YOUR IT CONTRACT:

- ✓ **PEOPLE/RESOURCE RISKS**
  - o Staff turnover; different corporate cultures, work ethics & cross-communication between parties
  - o Assigning proper number of appropriately skilled staff to the job when needed
  - o Wavering or lack of top management commitment; an unavailable IT Steering Committee
  - o No project champion(s)
  - o Unreliable referencing checking and reference checks

- ✓ **REQUIREMENTS RISKS**
  - o Poor project charter and definition of roles and responsibilities of the parties
  - o Ambiguous, incomplete, complex, contradictory requirements; Unspoken implied requirements
  - o System's "-ability" requirements not defined and/or not connected to performance metrics
  - o Building well-tested and fast interfaces to other products/systems.
  - o Cleansing, mapping and converting data from the old system to the new one.

- ✓ **TECHNOLOGY**
  - o Maturity level of needed: hardware, application and systems software, netware, and databases
  - o Scope and limitations of internet and mobile usage
  - o Security & Privacy: correctness, integrity, availability
  - o Automated tools to be used in developing/maintaining the system including:
    - ▪ Availability?
    - ▪ Cost?
    - ▪ Maturity?
    - ▪ Training required?
    - ▪ Usefulness and how it will be used and by whom?

- ✓ **PROJECT & TECHNOLOGY MANAGEMENT RISKS**
  - o Project Management:
    - ▪ Assuring "not rosier than actual" progress reporting?
    - ▪ Surfacing, identifying, documenting, resolving, escalating and managing issues, bugs, problems and disputes?
    - ▪ Managing differing constituent expectations: e.g., customers, departments, client, users, vendors, attorneys, customers, etc.
  - o Unclear Leadership – incl. Systems Integration Project Manager (SIPM) roles & responsibilities
  - o Fast escalation processes that lead to fast and appropriate decisions acceptable to all parties

- ✓ **PROCESS RISKS**
  - o Project underestimated or misestimated
  - o Managing staff productivity
  - o Properly memorializing the content, quality & "correctness/effectiveness" of interim deliverables
  - o Tradeoffs between cost, schedule, functionality & quality of systems development, implementation and maintenance
  - o Managing almost inevitable scope change
  - o Assuring enough time for adequate testing
  - o Taking shortcuts against the agreed to SDLC on the fly

- ✓ **PRODUCT & OTHER RISKS**
  - Performance, testedness & readiness
  - "–abilities" (scale-, use-, test-, port-, maint-)
  - Competent turnover & support processes -- developers to implementers to users to maintainers
  - Other risks (outside of the project)
    - Competitive risks
    - Economic risks
    - Organizational risks
    - Regulatory risks

IV. EXAMPLES OF IT CONTRACTS IN LITIGATION

Before launching onto the myriad clauses, I believe that it is instructive to briefly review a couple of cases where I have been an expert witness to better see just why and where the IT project fails and how a good IT contract might have prevented or mitigated such failures.

The following two example cases I have selected illustrate several major themes that recur in litigation in systems development projects. In reviewing such highly summarized cases, it is easy for the reader to underestimate the amount of know-how, sleuthing and preparation required. Once you know the answer, it's so simple -- but in some cases we have analyzed 500,000 -1,000,000 pages or electronic images of documents to arrive at and support our conclusions. With two or more sides to every story and even more contradictions, it takes hard work and tenacity, along with simple, independent, credible and persuasive testimony to the triers of fact to win the day.

Case 1: "The Good, the Bad, and the Ugly"

The System and the Parties

My first example case, from the early 1980s, began as a copyright and trade secret infringement case but it ended up turning on testing objectives. Sometimes you never know where the cases may take you.

In this case, one party, a small systems applications developer (the "Developer") built, licensed and maintained vertical software for the wholesale distribution business very early on when no appropriate solutions were available. Eventually he enhanced and globalized it until it became the very popular and robust industry leader. Now in his seventies, he wanted to sell the software company in a way that kept his employees gainfully employed within the acquiring company.

The opposing party (the "Acquirer") was a billion dollar systems developer and integrator that already enjoyed success with its own products – but in the manufacturing arena. The Acquirer believed it was presently losing customers who wanted integrated manufacturing and distribution package and thus wanted an outstanding complementary distribution product to complete its product line offering.

The Complaint

After looking at many available systems, the Acquirer gave the Developer a multi-million dollar letter of intent to purchase the assets and software of the latter's company, with the right, "in its sole judgment", to cancel the deal if after intense examination the software did not meet the needs of the Acquirer.

From preliminary discussions and visits the Acquirer knew about the organization and the software, its history, its pricing and costs, its competitors, its functionality and capabilities. The Acquirer claimed it now needed to review the Developer's software code, technical architecture and structure in detail, approve its design, interview the customers, programmers and testers, trainers, implementers, maintenance staff, hot-line group, marketing staff, and sales staff, review standards and documentation, and review and analyze the testing suite. The Acquirer would then apply its own system tests, to be sure that the system worked as promised and could be maintained by the Acquirer.

This all sounded reasonable at the time, given applicable law required the Acquirer act in good faith -- even though it could cancel the deal in its sole judgment. The letter of intent also provided that the intense review would be completed within 45 days. Well, Bang! On the 45th day exactly, the Developer received a one-page letter that indicated that the Acquirer would not complete the acquisition because the system did not meet the performance requirements promised by the Developer, or the profitability requirements needed by the Acquirer. The Acquirer in a very short time came out with a similarly configured system.

The Developer at first charged the Acquirer with copyright infringement. Following our initial involvement in the case as expert witnesses, and based upon our opinions, the attorney charged "bad faith" and "trade secret misappropriation" -- very difficult to prove!

The Analyses, Strategies and Opinions

During our research we discovered that the Acquirer was targeting the $2-$20 million dollar wholesaler distributor customer with particular focus on SIC codes 50 and 51. Tracking back to the actual business plan of the Acquirer, we also discovered in their mission statement that they said, "We are to get our product to market in a record 11 months -- and not the 23-26 months that we estimate to develop this system in-house from scratch."

Through detailed research regarding the average order dollar values and number of line items per order for relevant SIC code orders in that time frame, we determined that the highly "over-transactioned" performance tests performed by the Acquirer supported distribution companies with sales between $350 and $400 million in annual sales, 10 times larger than the Acquirer's targeted customers. We concluded that the tests were, in fact**, intended to make the system fail.** Fortunately, the actual speed and timing of the transactions were logged automatically on the system during these "load tests". This enabled us be very accurate in our analysis instead of having to estimate the transaction rate and response times.

With our analysis and opinion as their support, the Developer's attorneys argued that the tests were performed in "bad faith". This legal attack would not allow the Acquirer to avoid the letter of intent because of the results of the erroneous tests. We went on to prove that if appropriate loads were used, the system would have performed within the promised sub-second response time 98% of the time and never over a 2.5 second response time (vs. the unacceptable 20-45 seconds resulting from the Acquirer's loads).

Bad Faith

Next, we established motives for bad faith to show the jury why the Acquirer went through all this trouble and effort if it never intended to accept/acquire the software. We discovered that of the 6 people who reviewed the system in excruciating detail, 4 of them went on to develop the Acquirer's own system from scratch. And while their original estimates indicated it would take over 2 years to develop, in fact the system was running and available in slightly more than half that time.

We pointed out that through its intensive review and analysis of the system, afforded them under the letter of intent, the Acquirer was able to:

- Acquire the ability to bring their product to market more quickly by understanding how certain complex algorithms worked in the Developer's system
- Learn how the best available system was designed and implemented, including the use of standards, technical architecture and design considerations, screen navigation, levels of reporting, and module interfaces
- Adopt a suite of functionality that was studied and evolved by the Developer in over 17 years in the marketplace -- thus avoiding a long trial and error process in determining what current customers really needed and wanted (i.e., the Trade Secrets)
- Learn the future enhancements that were slated, and their estimated costs and benefits

Trade Secret Misappropriation

While it is very difficult to prove infringement of copyrights based upon similar functionality (because copyrights only protect the expression of the function, not the idea of the function), we were successful by employing unusual tactics. We argued that the particular set of available and future features and functions was indeed a trade secret of the Developer.

Furthermore, by copying this suite of functionality, even though with a different "look and feel", the Acquirer would be able to (1) bring its product to market more quickly, and (2) win away customers and prospects from the Developer -- two things that make trade secrets so valuable -- and that could only be accomplished by having access to other developers and documents (under false pretenses).

We were also able to bring into question curious similarities regarding interfaces between functions in both the Developer's and Acquirer's systems. While there were several "public domain" ways to implement any of the "allegedly copied" critical functions and algorithms, the Acquirer almost always did it similarly to the Developer -- an astronomically low probability for 12 complex functions if no copying occurred.

Lastly, we developed an analysis that showed that the Acquirer would be able to achieve the profit goals and margins targeted in its business plan for the new system because of the synergies provided by the Acquirer's manufacturing applications and the market position already enjoyed. We developed several launch, pricing, and maintenance scenarios to prove this point.

The Verdict

After four years of arguing this case, it settled almost literally on the courthouse steps, the day I was to testify, in favor of a happy Developer and his attorneys.

Case 2: "Shoes, Shirts, In-Stock, Out-of-Stock"

The System and the Parties

This 2003 case example deals with the implementation of a world-class "ERP" system for one of the world's largest retail clothing operations ("Customer") – who needed the system before fast approaching Y2K. The vertical retail version of the product was essentially "brand new" (i.e., like a beta) and would require, more than ever, dramatically increased customer-site installations, volumes of data, interfaces to third party systems, and challenging distribution networks. Former systems developers and consultants jumped on the bandwagon of the leading ERP companies to become "leading" integrators of those ERPs – and they had to trail-blaze many "firsts".

The Customer decided to license and modify the "best" (most expensive) integrated ERP solution, rather than buy smaller but still "best practices" leading application packages from different vendors, that they would then integrate. To avoid stock-outs while maintaining a reasonably low level of inventory, the ERP had to keep track of thousands upon thousands of products, sizes, fabrics and colors, so that the customer could forecast, know how much was sold, how much to order, how to price in relation to competitors in each market, when to mark down seasonal or slow moving products, and to distribute them efficiently and on-time to all of its one thousand plus stores. All of this required number crunching on a scale not previously tamed by an ERP.

Several other factors made this implementation a real challenge: 1) the project faced the hard and fast Y2K deadline 2) the customer wanted the "Cadillac" of ERPs, even though they faced severe budget restrictions, and 3) the customer wanted to do much of the work with its own staff to save money, even though that staff lacked the experience and qualifications for the job. With no time to run a proper competition for an integrator, the customer turned for guidance and advice to the consulting arm of its auditor ("Consultant") and proceeded without properly formalizing who was responsible for what.

After several schedule slips, the project squeaked by the deadline and went live. However, the customary period of bumpiness following any ERP implementation took longer than usual, and the customer was unhappy about the consequent loss of profits and consumer goodwill and subsequent stock price drop.

Prelude

The customer had already successfully sued the ERP developer with whom the customer had its own contract, not involving the Consultant. The system was missing several critical functions at go-live, and many aspects of the new system could not be adequately tested in the resulting compressed time frame.

The customer was not satisfied with the recovery in this settlement, and so it turned its sights on the Consultant that it asked to guide the project.

The Complaint

In the complaint, the Customer alleged that the Consultant was hired as a general contractor-type integrator, responsible for all aspects of the project and responsible for delivering a working system on time. The customer alleged that the Consultant, abusing its audit relationship of long standing, did not have the qualifications or staff needed to successfully manage the project and deliver the working system on time. Further, the Customer alleged that the Consultant failed to give proper advice about hardware requirements, functionality, and how to manage the developer. The Customer alleged that the Consultant failed to properly test the system before going live. Ultimately, the Customer blamed the problems experienced after go-live on bad design and configuration, which it blamed on the Consultant.

The Analyses, Strategies, and Opinions

Hardware: It was evident from the outset that the implementation of the ERP would needed lots of processing power to handle all of the number crunching required within the limited nightly batch window. There was a deep concern about what kind of hardware would be required to handle the very data-driven US retail market. No configuration this large had ever been developed and used. A preliminary stress test and performance test was prepared by the developer and reviewed with the Consultant and Customer. The test, which by necessity was based on a smaller operation, raised serious concerns about how much hardware and computer horse power would be required to assure that the 1,200 stores could be polled and processed daily, analyzed, and turned into inventory and transportation requests that would replenish goods at each of the stores before opening time the following morning. Failure to meet the daily turnaround schedule would be catastrophic, not just inconvenient. Accordingly, both the consultant and the developer made strong recommendations from the outset that the customer should purchase the most powerful servers then available on the market at that time. Despite those recommendations, the customer opted to buy servers from its current hardware provider, based on that provider's unsubstantiated promises about throughput performance, and based on the fact that this option was cheaper than the purchase of more powerful servers. We were able to determine that the hardware selected caused a number of problems on the project. First, it caused delays, as it took much longer than expected for the hardware vendor to pass the preliminary performance stress test, even to make crucial decisions and trade-offs about how to design and configure the ERP functionality.

Authority and Control: A major thrust of the customer's case was that the Consultant was a general contractor-type integrator, and much of the opinion of their expert witness was premised upon that allegation. Among other things, we opined that a general contractor-type integrator typically has authority to: determine what tasks need to be completed; define a schedule of deliverables and deadlines; determine how many personnel and who to assign to the various tasks; hire and fire subcontractors on whose deliverables the integrator depended; and sanction or make changes to address poor quality or tardiness of deliverables. We showed through contract and meeting documentation that the Consultant did not have such control or authority. The plaintiff's wish to hold the Consultant responsible without giving it authority to manage to succeed was not a defensible position.

Project Management: The Consultant played a major role in the planning the work. Senior managers of the Consultant were assigned as project managers, and these managers created project plans for the various teams. We were able to show that the customer, however, retained authority over those plans. We were also able to show through meeting and planning documents that the customer retained control over the work, deciding which resources would be assigned to do what work and deliverables, and in what sequence. We explained the importance of the fact that the customer established direct relationships with the other system and project vendors, and how that prevented the Consultant from exercising any degree of influence over work that was affected by those other vendors.

Qualifications: The customer alleged that the Consultant concealed its lack of personnel after promising highly qualified people. We were able to demonstrate that the customer understood the risks in being one of the first implementations of the retail ERP in North American, and that the fallout was that fewer resources were available with prior experience in implementing that solution. We showed those conditions are inherent in any pioneer project. We opined that the Consultants assigned personnel had sufficient qualifications for the work, and adequate transferable experience and expertise from predecessor versions of the system and implementations.

Gaps: Initially, the customer engaged the Consultant to identify a few hundred functionality gaps in the developer's solution. Thereafter, the customer excluded the Consultant from the process of managing and resolving those gaps. The customer established its own contract with the developer, and itself managed the developer's work on the gap solutions. We were able to demonstrate that the Consultant bore no responsibility for the poor quality of the gap solutions and the delays in their completion. We were also able to explain how the delays in the gap solution caused other delays in the project, including the extension of the design phase, and the consequential compression of the testing phase.

Scope: As in most projects, the customer's business personnel played a crucial role in the design, configuration and testing of the system. Documents that we researched and analyzed revealed the customer's business experts were stretched too thinly, often unavailable to make timely decisions, often disagreed with one another, and revisited and changed decisions they had previously made. We were able to explain the impact that this has on the design, and the wasteful need for more time and effort on re-working and re-testing modules on the system.

Testing: From the outset, the Consultant insisted that one of the Customer's experienced staff be appointed as "Test Integration Manager" (TIM) to oversee the testing of all components and integrating them into a cohesive whole. Only very late in the project did the customer accept this recommendation. We were able to show that the Consultant's TIM did the best under the circumstances to get the most important testing done within the very limited time frame left. We explained that the Consultant could not be blamed for overlapped and incomplete testing, where that incompleteness resulted from delay caused by the customer's conduct and choices. We also opined that, under the circumstances, the Consultant did an admirable job in adapting its methodology to cover the crunched testing schedule which, in fact, contributed to the success of this project.[9]

Poor Communication: The Customer lambasted the Consultant in the litigation over the Customer's perception that the Consultant did not adequately communicate its recommendations and observations about project risks. However, Consultant managers testified that they communicated all of their concerns and observations about project risks verbally to the Customer's managers on an on-going basis, and to customer top executives (President and CFO) as needed. We were able to explain that the optimal mode of communication often depends upon the particular circumstances of a project, and the personalities involved.

Post Go-Live Problems: The ERP system did go live with limited functionality, due to a number of pending crucial gap solutions. The system did experience some stress and bumpiness with the customer blaming the problems on design decisions made by the Consultant, and inadequate testing of system components. We were able to show that the problems emanated in part from the underpowered hardware, and that once the hardware was upgraded most performance problems vanished.

We opined that several other factors hampered the functionality, including the late implementation of a strong archiving procedure to offload the more than 10 gigabytes of daily detail data to a summarized data warehouse, originally and repeatedly recommended by the Consultants but refused by the customer until well after go-live.

Moreover, we were able to demonstrate that some problems could have been avoided if the customer had followed the performance tuning recommendations that the Consultant made prior to go-live. We demonstrated that many problems related to the customer's own choices, such as lack of proper training of users, poor qualifications of customer technical staff, poor configuration of the customer's data center and a poor conversion process conducted by the customer that led to initial data and reported stock-out problems.

The Verdict

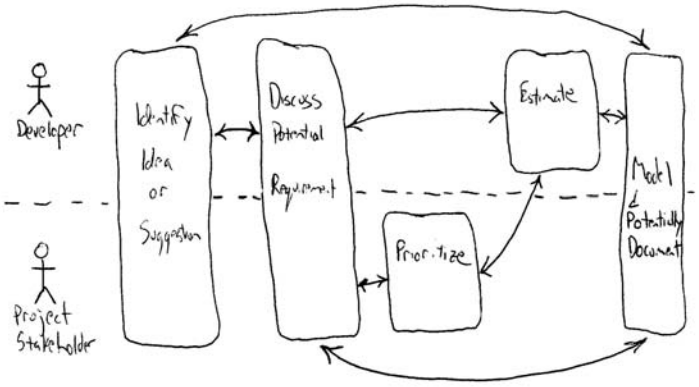
The Customer's expert issued a rebuttal expert report, which went noticeably on the defensive. Several months later, the court made a ruling on the Consultant firm's motion for summary judgment that devastated the Customer's case, and a few months after that, the customer walked away from the litigation.

[9] Note: we opined that this system/project was a success. Additionally, one year after go-live and extended bumpiness, the customer began achieving the bargained for benefits and told its investors and analysts so.

IV. AN INDEX OF IMPORTANT IT CONTRACT CLAUSES & RELATED THOUGHTS[10]

A. Thoughts On Functional Requirements & Performance Measures

This section of the contract identifies and memorializes the systems and documentation that are contracted for, and how they will look, be used, operate, and perform, and the reports, user interfaces, navigation, controls, screens that will be made available and/or produced as a result of user inquiries.



Note that it is critical to get requirements correct, complete and as unambiguous as possible from the start. Why? Because making changes to systems based on wrong or incomplete requirements can cost as much as 10 times to repair after it is designed with the wrong or missing, 25 times as much after it is programmed, 50 times a much after user training and conversion performed, 75 times as much after testing and acceptance, and 100 times after the software is in production.

Have information, responses and promises the vendors (or users) made during the RFP and selection process, at demonstrations, at User meetings, in its published materials TO THE EXTENT THAT YOU RELIED ON THESE THINGS, made part of the Contract. If the opposite party refuses to do so, find out why and how it might affect your revised selection results. If it is important, is the desiring party willing to pay or trade for it?

As important are Function Requirements to a customer achieving the business benefits it has planned for with the new system, so are the Performance Requirements. I call the performance requirements the systems "-abilities" because so many of them end with that suffix. Each of these –abilities must have SMART (Specific, Measurable, Attainable, Relevant, Trackable) metrics and measurable results associated with them, and remedies for failure to meet these measurable requirements.

Performance requirements may include: availability, reliability, flexibility, portability, scalability, Integrate-ability, dependability, operability, deploy-ability, modifiability, trace-ability, secure-ability, etc. Customer MIS, quality assurance, technical project personnel, and user department management and staff can help Counsel come up with SMART metrics.

3. FUNCT REQUIREMENTS & PERFORMANCE MEASURES (B)

3.1 Description of business features & functions to be performed (or x-ref to RFP)

3.2 Requirements Elicitation (RE) process & Scope Change Process

- Identify specific Users/Classes; assure/schedule availability
- Prepare timeline; Identify RE techniques/tools/docs used/produced
- Escalation/dispute resolution process
- Scope change process: proposed, estimated impact, decision criteria, documentation, update estimates, Critical Path Method, staffing, monitoring progress
- Embrace change: Iterative change, Incremental change, Interactive change

[10] **For your convenience, the colors of each of the contract sections below match those used in the IT Contract Success Model and related PowerPoint slides.**

3.3 Incorporation of Functional Requirements Documentation

- Requirements including: success criteria; '-abilities'; current & future Features & Functions; User Interface; reports; business processes & rules; operation parameters; Documentation map; Requirements Traceability Matrix (RTM) Gap analysis & changes required; rollout plan; test results; system stability; Go-Live checklist; requirements change process/tools; defer/tradeoff; Training plan; converted data; legacy Interface; security Passwords; enabled and trained Operations and Maintenance Groups, etc

3.4 Incorporation of Vendor's proposal(s), website, marketing materials, system demonstrations, Conference Room Pilots, etc.

3.5 Performance Parameters:

a. Relationship to functional processing requirements
b. Types & volumes of data to be stored
c. Number of users; Locations
d. Special features or capabilities
e. Hosting, SaaS, other service act w trial period
g. Use by affiliates
h. Transfers to other CPUs or locations
i. Right to make copies, including backup or archive copies
j. Growth capacity measured in increased transaction processing rates
k. Capacity for field modification or enhancement
l. Other Vendor representations

3.6 Performance "-ABILITIES"

a. Software QA: ['-abilities'] hardware, software, netware, architecture

b. USERS (External) care about these '–abilities':

Avail- Depend- Flex- Integ-
Oper- Reli- Modif- Use- Scal- Safety Secur-
Surviv- R&R- Config- Utility; Cost Effectiveness [vs benchmarks]

c. DEVELOPERS (Internal) care about these '–abilities':

Adapt- Audit- Deploy- Reuse- Test-
Interop- Maint- Perform- Port- Extense- Struct- Manufact- Trace-
Reuse- Test- Struct-

3.7 Relationship to acceptance criteria & testing

3.8 Relationship to ongoing maintenance requirements

B. Thoughts On Project Management (see http://communityharvest.com/project%20lead.htm)

Responsibility for the project

"Responsibility" is a hazy word with many meanings to different people. The definition of responsibility in the Concise Oxford English Dictionary is "authority; the ability to act independently and make decisions." This is, in effect, the first criteria of a project manager--can s/he independently make decisions regarding the project, without recourse to a higher authority within the overall envelope of approved time, cost and scope, ***and*** change strategy, approach or activity definitions to accomplish the same results within the same window.

Accountability for project results

Project managers are accountable for the results of the project. This may or may not equate with the delivery of the full business case of the project – because for many systems projects, for example, management of the organizational change associated with the results of a project may not be within the scope of responsibility of the project team. Thus, whatever *is* within scope must be clearly defined and measurable, and the project manager must be fully accountable for the delivery of those results.

Authority to execute the project to get the results

Project Managers must have the authority to execute the project in order to realize the intended results. Authority can be defined as "the power to enforce or influence behaviors or actions necessary to attain the results for which they we are being held accountable."

While the project manager may have the responsibility to make decisions *inside* the project, and the accountability for the results of the project, that person must also have the ability on behalf of the organization to ensure availability of resources and require the changes of behaviors that are necessary. In other words, without authority, a project manager cannot be held accountable or responsible for the management of a project.

Based upon these attributes, a reasonable definition of project management is *"The exercise of responsibility and decision-making about a project, the authority to execute within the boundaries of the project, and the accountability to deliver the results of a project in the context of agreed-upon customer expectations, commitments and constraints."*

To be held accountable for results, project managers must have the responsibility to make decisions about a project or the authority to attain results

The implications of this definition are potentially far-reaching. Project Managers who do no have or do not exercise their mandate with responsibility *and* authority *and* accountability typically are the managers of failing projects.

Note that there are many tools, techniques and practices used in the proper management of successful projects.

It is beyond the scope of this thought piece to get into them, but more information can be found at the following URL's among others:

http://www.pmi.org/Pages/default.aspx

http://www.sei.cmu.edu/cmmi/

http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Giga_-_Project_Management_Best_Practices-_Key_Processes_and_Common_Sense_%281-03%29.pdf

http://www.stsc.hill.af.mil/crosstalk/2004/10/0410Jones.pdf

http://project-management-software-review.toptenreviews.com/

PROJECT MANAGEMENT (C)

4. PROJECT TIMETABLE (C)

4.1 Definition of project tasks
4.2 Definition of deliverables for each task
4.3 Estimating methodology used; assumptions; update process
4.4 Customer's responsibilities; Vendor's responsibilities
4.5 Target completion dates by task
4.6 Final completion dates by task

4.7 Customer's right to delay or cancel project tasks
4.8 Major perform milestones & relationship to payment schedule
4.9 Delay remedies & bonuses for early perform
4.10 Relationship to termination rights
4.11 Prompt notice of anticipated delays

5. PROJECT MANAGEMENT (PM) & REPORTING (C)

5.1 Steering Committee's role, structure, makeup, processes
5.2 SWAT (SoftWare Adjudication Team) Team's role, structure, makeup, processes
5.3 Named system Integrator's role, responsibility, authority, accountability
5.4 Description of Vendor's project team; Description of Customer's project team
5.5 Designation of Vendor's Project Management process; Designation of Customer's Project Management process
5.6 Determine SDLC METHOD (Software Development Life Cycle Methodology) to be used; rules, documentation, risks and approval processes for deviating from SDLC
5.7 PM Tools/Measures: Earned Value Method (EVM); Critical Path Method (CPM); Estimates-To-Complete (ETC); coat and schedule variances
5.8 Vendor's responsibility for PM; project reporting: format; frequency; distribution list; detail
5.9 Vendor's/Customer's responsibility for: Business Process Reengineering (BPR) -- new process development, linking new processes, staff training, process testing, approval(s), cutover
5.10 Vendor's responsibility to ID, manage, mitigate RISKS, problems, delays
5.11 Customer's responsibility to address Vendor's problems/recommendations/issues contained in the Project Reports
5.12 Customer's responsibility to assist Vendor
5.13 Customer's responsibility for project problems or delays
5.14 Relationship to project timetable
5.15 Remedies for loss/reassignment of Vendor's Project Manager (PM)

12. PROJECT COSTS & PAYMENT SCHED. (C)

12.1 Hardware prices, if purchased (including "bundled" software)
12.2 Software prices, if purchased (and to be stated separately)
12.3 Rental or lease payments & method of calculation
12.4 License fees for software
12.5 Training fees
12.6 Maintenance fees for equipment &/or software
12.7 Fees or charges for other services
12.8 Partial payments tied to major milestones; % holdbacks at milestones until whole system is accepted
12.9 Start date for rental or lease payments
12.10 Commencement date for license fees
12.11 Commencement date for maintenance fees
12.12 Credits or offsets for delays or failures
12.13 Refunds if Contract is terminated

12.14 Most favored nation clause
12.15 No modifications or additional charges w/o written approval
12.16 Invoicing procedures
12.17 Supporting documentation
12.18 Required notice for price increases
12.19 Personnel/services rates to price change orders
12.20 Limits on price increases
12.21 Right to benefit of price reductions
12.22 Relationship to Contract remedies
12.23 Payments under protest
12.24 Offset rights
12.25 Relationship to dispute resolution mechanisms

C. Thoughts on Go-Live Readiness

Many tasks must be completed and questions answered before a system is ready to go live.

The sites must be made ready. New servers and wireless devices may have to be put in and tested. Physical and electronic controls must be implemented. New authorizations for use and access, and passwords must be created. New cabinetry and desks may be needed.

The equipment must be configured to work at the site and with the other equipment and software it must interface with. New version of the operating system and the utility software will probably need to be installed and tested. Diagnostic test errors will have to be addressed. More hardware may be required.

The new applications must be loaded onto the new hardware, software and netware. Typically, multiple parties have responsibilities for getting the site ready and the appropriate software loaded, tested and accepted.

Which users will have to be trained? Will it be a “train the trainers” approach? Which MIS operations staff will receive training? Which help desk people? Which training must each class of users take? Where will the training be held? How long is each training session? How often? Is refresher training available? Will the trainees be able to “play with the system in a non-production training environment” back at their offices until the system goes live? Is their computer based training available? Is the level of documentation appropriate for your mix of users?

What documentation will be delivered? Training? Reference? Troubleshooting? Systems design documentation? How will it be delivered and updated? To whom and by whom? Will help screens be available to users? Will the requests for help be context sensitive? Is the level of documentation appropriate for your mix of users?

Have all the appropriate tests been performed? Are the results satisfactory? Is there an acceptable level of known but still unfixed errors in the system? Is the help desk in place? Are the trained business function and technical staff, processes and automated tools ready to address the problems that invariably occur during the settling period after Go-Live? Are the vendor/developer/consultants who worked on the systems development and/or implementation of the system available and on-site to address warranty issues, defects, errors and fixes that will come up in the first 90+ days? Have current contact lists been distributed? Are backup, recovery and restart procedures in place?

Have the new workflows, policies and procedures, reporting relationships and job descriptions been activated? Is the staff ready for the cutover? Has the new software application system been tested in conjunction with the staff and procedural changes needed to support the new systems capabilities? Are outside third parties trained and ready for their new interfaces with the new system (e.g., suppliers, customers, outsourced vendors providing services, etc.)?

PRE-GO LIVE TASKS

6. SITE PREPARATION

6.0 Identify of general contractor; roles and responsibilities; liens; overall schedule
6.1 Preparation & delivery of site prep specifications
6.2 Customer's (or Vendor's) obligation to prepare site
6.3 Vendor's obligation to clarify specifications
6.4 Vendor's obligation to inspect & certify
6.5 Remedies for improper site preparation
6.6 Remedies for inspection errors
6.7 Effect on project timetable

7. COMPUTER CONFIGURATIONS DELIVERY & INSTALLATION

7.1 Delivery of complete equipment configuration
7.2 Delivery of operating system & other system sw
7.3 Access to site
7.4 Installation obligations of Vendor
7.5 Installation obligations of Customer
7.6 Diagnostic tests & relation to Acceptance Test provisions
7.7 Definition of completion of equipment installation
7.8 Remedies for delays in delivery or installation
7.9 Relationship to termination rights

10. TRAINING

10.1 Vendor's obligation to provide training
10.2 Qualifications of trainers
10.3 Location of training
10.4 Standards for acceptable performance
10.5 Relationship to timetable
10.6 Relationship to project costs
10.7 Availability of student materials
10.8 Availability of instructor's materials & training
10.9 Number of trainees
10.10 Buyer's rights to reproduce & use training materials
10.11 Continuing availability of standard Vendor classes
10.12 Continuing availability of on-site training by Vendor
10.13 Remedies for delays in providing suitable training
10.14 Availability and cost for refresher training (based on whose fault?)

11. DOCUMENTATION

11.1 Description of types of docs
11.2 Documentation standards for user, system & programming documents
11.3 Relationship to project timetable
11.4 Relationship to performance measures
11.5 Customer's rights to reproduce docs
11.6 Customer's rights to future docs or enhancements
11.7 Customer's rights to source code & related docs

11.8 Remedies for delays or inadequate docs

14. LOW LEVEL, SYSTEM & INTEGRATION TESTS

14.1 Diagnostic tests of hardware, infrastructure, netware
14.2 Different for different kinds of systems:
End-User; MIS; Outsourced; Commercial;
Military; Operating system (latent error statistics available)
14.3 All manner of tests for functionality & '-abilities' including:

DYNAMIC TESTS – can uncover only up to 85% of the defects and include:
- **white/black box; sub-routine; unit; integration; systems**
- **Interface, data audit, test of/w converted data**
- **new function; regression; performance; capability;**
- **Independent V&V; Viral, Security; Acceptance; Beta**
- **'-ability' (use', scal-, port-, maint-, reuse-, recover-, bullet, and many more**

STATIC TESTS – can uncover many of the errors that dynamic testing doesn't (up to 15%). These tests include requirements, design, code and test:
- **reviews**
- **walk-throughs**
- **inspections**

14.4 For each test level consider number of tests and:
- **Goal, objectives, success criteria**
- **Definition of accept results to move to next test level**
- **Test tools: decide, acquire, train; # test environments**
- **Experience needed for test staff: test leads, SMEs, analysts, testers, QA**
- **Test: scheduling, training, environment, equipment; access**
- **Test results review: who, when, turnaround time**
- **Number of shifts; Definition of scenarios; end-to-end tests**
- **Standard test docs; signoff; required; rigor, format & detail; error logs**
- **Rigor of error root cause identified, estimate to fix, fix process**
- **Test metrics to measure & report progress, issues and problems**

14.5 Description of error severity levels and time to repair each
14.6 Escalation Process; SWAT - software Adjudication Team
14.7 Maintenance service during testing
14.8 Relationship to performance measures
14.9 Remedies for failure to meet test criteria
14.10 Relationship to termination rights

15. ACCEPTANCE TESTING

15.1 Live or simulated environment? With new business process (resulting from Business Process Reengineering (BPR) project)
15.2 Description of test data and responsibility for its preparation
15.3 Description of test procedures & criteria
15.4 Relationship to functional requirements & performance standards
15.5 Period of testing
15.6 Review of test results
15.7 Correction of errors & problems
15.8 Definition of acceptance
15.9 Remedies for failure to meet accept criteria
15.10 Relationship to warranty & maintenance provisions
15.11 Relationship to termination rights

D. Thoughts on Customization and Data Conversion

One of the most overlooked and least understood aspects of a large systems implementation is the impact of **custom programming** and the double-edged sword of **data conversion.**

Let's make sure we know the difference between **customization and configuration** of software.

Let's start with a software package like: SAP; or PeopleSoft; or Retek; or MACPAC; or Oracle's e-commerce suite; or MicroSoft's word. These application programs were developed for sale to the general public. Packaged software is generally designed to appeal to a large audience of users, and although the programs may be tailored to a user's taste by setting various preferences (the process called "configuring" or "configuration"), it is somewhat limited AND NOT as individualized as custom-designed and custom-programmed software. Custom software is specifically designed and programmed for an individual customer in contrast with a software package.

Configuration involves making choices about which features and functions already programmed into the system to engage, and what options and settings to select in doing so. Some configuration of large software packages may actually look like programming in that they are undertaken through specialized proprietary software languages.

Customization involves adding new features and functions that were not already present in or contemplated by the software package. This can be accomplished by writing entirely new computer code, or by "bolting-on" other software packages. Well thought-out software packages actually provide "places" for new computer code and other software packages to latch onto the software package (through Application Programmer Interfaces – i.e., APIs).

Custom programming brings the added challenges of requiring a more rigorous testing and interface requirements, as well as potential difficulties when the core system gets updated by the vendor – without considering the impact the customized software may have on it. Usually the more rigorous testing and a regression testing suite is required every time an update to the core package is made.

Data conversion is changing data from one file or database format to another. Information system conversion requires data conversion and either program conversion or installation of newly purchased or created applications.

The **conversion process** involves many complicated and meticulous steps. User data typically must be cleansed to eliminate special uses of data fields and old, blank and/or erroneous data before the old data can be extracted, mapped, loaded, tested and audited on the new system. In my experience, this process is either started too early or too late and is typically misestimated. The contract must help assure that this process is done right, at the right time and results in a complete and properly linked database. Responsibilities must be carefully distributed and progress must be frequently reported. This is the area where finger pointing between the parties can be rampant.

CUSTOMIZE & CONVERT

8. CUSTOM PROGRAMMING SERVICES

8.1 Incorporation of functional processing requirement docs
8.2 Vendor's obligation to develop custom apps
8.3 Relationship to stipulated performance measures
8.4 Relationship to project timetable
8.5 Development by Vendor of detailed design specifications
8.6 Review/approval by Customer of detailed design specifications
8.7 Specification of programming standards
8.8 X-reference to system, program, and user docs standards
8.9 Specification of change order procedures

8.10 X-reference to system & acceptance test provisions
8.11 Vendor's responsibility for acceptable unit/system test procedures
8.12 Vendor's obligation to deliver source code & related documents
8.13 Relationship to project timetable
8.14 Remedies for delays in completion
8.15 Relationship to termination rights

9. CONVERSION & OTHER SUPPORT SERVICES
9.1 Data conversion
9.2 Application program conversion
9.3 Development of test data
9.4 Assistance to Customer with site acquisition &/or prep
9.5 Assistance to Customer in acquiring other products/services
9.6 Assistance to Customer in locating & screening employees
9.7 Coordination of telecom procurement
9.8 Responsibility for trouble-shooting
9.9 Assistance to Customer with development of backup plans
9.10 Assistance to Customer with backup arrangements
9.11 Assistance to Customer in developing security plan
9.12 Assistance to Customer in developing disaster recovery plan
9.13 Pre-installation machine time

E. Thoughts on Personnel

In so many cases the reasons for failure often boil down to whether the right people staffed the project pieces at the right time. It's not so much about IBM, HP, the Internet, Microsoft Project, mobile devices, or network technology…

As it is about the people:

1. The ones who understand the company's business, direction and culture who can make good decisions about the systems that are needed to support them
2. The ones with the right systems, management, technology experience and expertise
3. The ones that can lead a project and a project team
4. The ones that know the customers' industry
5. The ones that have configured the software and know its limitations
6. The ones that have implemented it before (in similar organizations, if possible)
7. The ones that know the weaknesses of the system – and how to overcome them where needed
8. The ones who understand the right SDLC to be used

Remember, during the project, the needs for different staff and skills will change as the project progresses over time. Make sure the appropriate people are assigned to the project and made available **when** they are needed!

1. Project Estimators and administrators
2. Subject Matter Experts (SMEs)
3. Requirements elicitors
4. Analysts
5. Designers
6. Programmers
7. Database experts
8. Test scenario developers, testers, debuggers
9. Systems tuners
10. Documentation specialists
11. Trainers
12. Maintenance staff

The contract is very important here because not having the right skills at the right time can cripple a project. Discovering bad work and deliverables many steps later in the project after they were actually introduced can cost orders of magnitude of increased dollars (10 – 100 times) to identify, isolate, repair, retest, perhaps retrain, and release the fix, than if caught at inception or avoided by the right people and processes altogether.

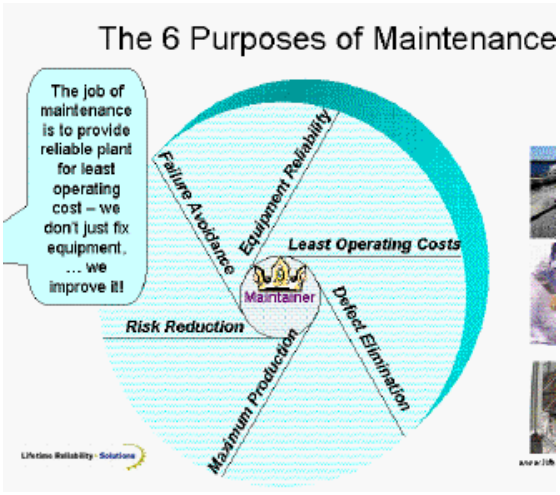
The following list covers many of the personnel issues that should be on the table and in the contract to help mitigate the problem inherent with staffing and personnel.

13. PERSONNEL

13.1 Vendor's staff qualifications/Customer's approval rights; evaluations; courses; experience
13.2 Application certified/certificates; Updated CMMi rating
13.3 Periods of Availability
13.4 Prohibitions against interruptions in Availability
13.5 Temporary replacements for sickness, etc.
13.6 Right to request replacements
13.7 Prohibition against removal or reassignment
13.8 Ability to pass on Vendor's staff salary raises to Customer

F. Thoughts on Warranties And Maintenance

Warranties:



Carl D Howe, Principal at Blackfriers Marketing, asked in 2003: "**Would you buy the car described below?**

> **FOR SALE:** ***Brand new 2003 Mercedes Benz S600, Brilliant Silver, Charcoal Grey interior with COMAND navigation computer, ABS brakes, ESP. Seller disclaims all warranties and conditions, either express, implied or statutory, including, implied warranties or conditions of merchantability, or fitness for a particular purpose with regard to the car. The entire risk as to the quality of or arising out of use or performance of the car and its support services, if any, remains with you. Price: $125,000 firm.***

Most buyers would ignore this offer. Who would pay over $100,000 for a car that the seller won't guarantee can at least drive off the lot?

Yet these are the exact software licensing words that enterprise software buyers have been agreeing to over the last 20 years or so. And corporations have been paying a lot more than $100,000; in many cases, they've bought software for millions of dollars with the same terms. Worse, most software companies also charge mandatory maintenance fees, often just to fix defects in the products they never should have shipped in the first place."

Warranties have typically been tricky and difficult to accept in the software industry. **Microsoft's End User License Agreement (EULA) is particularly one sided (below)** – but it hasn't seemed to hurt its' sales.

Sap and Oracle, while perhaps less offensive that Microsoft's shrink-wrapped licenses also have strict and limited warranties. It is possible to negotiate changes to these clauses in certain cases with the ERP vendors if you are willing to make other tradeoffs.

It is beyond this thought piece to detail re such cases, but warranties are important and you must understand the risks you take/or want the other party to take. Be prepared to pay to mitigate such risks.

From Microsoft's End User License Agreement (EULA) is particularly one sided:

> 14. LIMITED WARRANTY FOR SOFTWARE ACQUIRED IN THE US AND CANADA. Microsoft warrants that the Software will perform substantially in accordance with the accompanying materials for a period of ninety (90) days from the date of receipt. If an implied warranty or condition is created by your state /jurisdiction and federal or state/provincial law prohibits disclaimer of it, you also have an implied warranty or condition, BUT ONLY AS TO DEFECTS DISCOVERED DURING THE PERIOD OF THIS LIMITED WARRANTY (NINETY DAYS). AS TO ANY DEFECTS DISCOVERED AFTER THE NINETY-DAY PERIOD, THERE IS NO WARRANTY OR CONDITION OF ANY KIND. Some states/jurisdictions do not allow limitations on how long an implied warranty or condition lasts, so the above limitation may not apply to you. Any supplements or updates to the Software, including without limitation, any (if any) service packs or hot fixes provided to you after the expiration of the ninety day Limited Warranty period are not covered by any warranty or condition, express, implied or statutory. **Maintenance** is critical to the life, usability and value of any system. I have witnessed more than one big company pay a fortune for a large ERP system implementation, only to see it go bad in less than 1 year because it was not properly maintained.
>
> 16. DISCLAIMER OF WARRANTIES. The Limited Warranty that appears above is the only express warranty made to you and is provided in lieu of any other express warranties or similar obligations (if any) created by any advertising, documentation, packaging, or other communications. Except for the Limited Warranty and to the maximum extent permitted by applicable law, Microsoft and its suppliers provide the Software and support services (if any) AS IS AND WITH ALL FAULTS, and hereby disclaim all other warranties and conditions, whether express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of reliability or availability, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the Software, and the provision of or failure to provide support or other services, information, software, and related content through the Software or otherwise arising out of the use of the Software. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THE SOFTWARE. See http://www.microsoft.com/windowsxp/eula/home.mspx

Maintenance:

There are several different kinds of maintenance:

(1) Hardware maintenance is the testing and cleaning of equipment.

(2) Information system maintenance is the routine updating of master files, such as adding and deleting employees and customers and changing credit limits and product prices.

(3) Software, or program, maintenance is the updating of application programs in order to meet changing information requirements, such as adding new functions and changing data formats. It also includes fixing bugs and adapting the software to new hardware devices.

(4) Disk or file maintenance is the periodic reorganizing of disk files that have become fragmented due to continuous updating.

Most maintenance contracts are actually separate contracts from the software license, software development, or systems implementation contracts. However, they are closely related to those in that without them being properly written, managed and evaluated against contractual service level objectives, the buyer will never achieve the longer range benefits it bargained for or ROI it planned. Maintenance oftentimes runs approximately 17% of the cost of the software license, development costs and hardware costs annually. It is alleged, but not surprising that companies such as Oracle now make more money every year from warranty income than from new sales.

Good, reliable maintenance is so important to achieve the planned goals of the new systems that time must be devoted/allocated to it during the reference checks & reference check site visits.

Important contract clauses for warranties and maintenance should address:

16. WARRANTIES (G)
16.1 Vendor's financial condition
16.2 Hardware warranties
16.3 Software warranties
16.4 Service warranties
16.5 Pass through of third party warranties
16.6 Relationship to performance measures
16.7 Start date(s)/length of warranty period(s)
16.8 Relationship to maintenance provisions
16.9 Scope of warranty obligations
16.10 Remedies for failure to meet warranty obligations
16.11 Assignability of warranties
16.12 Relationship to disclaimers & Limits of Vendor's liabilities

17. MAINTENANCE (G)
17.1 Start date(s) & length of maintenance period(s)
17.2 Vendor's termination rights
17.3 Required notice for termination
17.4 Customer's renewal rights for guaranteed period
17.5 Relationship to performance measures
17.6 Types & Description of maintenance support
17.7 Notice of defects or problems
17.8 Classification of types & criticality of maintenance problems
17.9 Required dispatch or respond time(s)
17.10 Escalation of maintenance support if delays to correct problems
17.11 Maximum repair time
17.12 Uptime guarantees
17.13 Replacement of "lemons"
17.14 Avail of spare parts or components
17.15 Limits on Vendor's refurbishment rights
17.16 Customer's rights to perform maintenance
17.17 Customer's rights to maintenance manuals
17.18 Customer's rights to maintenance training
17.19 Limits on Customer's rights to perform maintenance
17.20 Remedies for delays in providing adequate maintenance
17.21 Vendor's obligation to coordinate third party maintenance
17.22 Customer's right to get maintenance o/s principal maintenance period
17.23 Limits on increases in maintenance fees
17.24 Backup equip avail during extended maintenance periods
17.25 Customer's rights to future enhancements
17.26 Customer's rights to assign maintenance rights

****

For more information visit WSR Consulting Group's website @www.wsrcg.com
or call Warren Reid @ 818-986-8842

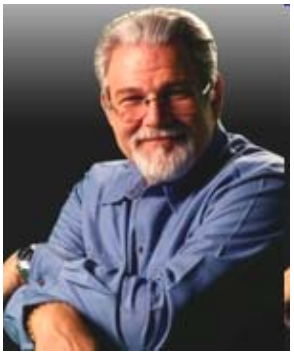
For full size views of the "IT Success Models" go to:

- http://www.wsrcg.com/PDFs/model_riskipedia.pdf
- http://www.wsrcg.com/PDFs/model_requirements.pdf
- http://www.wsrcg.com/PDFs/model_itcontracting.pdf
- http://www.wsrcg.com/PDFs/model_cpritprojturn.pdf

For more about our work as expert witnesses in IT Contract & System Project Failure Litigation, see:

- http://www.wsrcg.com/Articles/itj0511_reid.pdf
- http://www.wsrcg.com/Articles/ComputerandSystemsFailureLitigationQAI.pdf

Biographical Sketch for Warren S. Reid



Managing Director, WSR Consulting Group, LLC

Tel: **(818) 986-8846** *e-mail:* **wsreid@wsrcg.com** *website:* **www.wsrcg.com**

Following a distinguished career as a management and computer technology consultant and partner at an international consulting firm where he designed and implemented dozens of systems, and consulted on more than 50 IT contracts of all kinds (including software development, systems implementation, software maintenance, professional services, hardware procurement, outsourcing and more), Warren S. Reid founded the **WSR** *Consulting Group*, LLC in 1988. He has been engaged in developing and implementing large-scale systems and turning around systems in crisis situations, such as helping create and launch the Federal Energy Office for President Jimmy Carter in 75 days, overseeing the testing and acceptance of California's Lotto Lottery games in just 100 days, and helping to resurrect the Malaysian MESDAQ Lottery which failed on opening day.

Mr. Reid has extensive experience, expertise and knowledge in several industries including:

- The retail industry (multi-national retail & grocery store chains; international fast food restaurants)
- POS systems of all kinds (including lottery systems, movie and legitimate theatre ticketing and concession systems)
- E-business and e-commerce systems
- Health care, hospital and HIPAA systems
- Robotics manufacturing and smart buildings
- and more

He has been recognized as an expert and expert witness, and has testified in matters involving:

- The root causes of system failure
- IT project/software development & implementation estimating, scheduling, resourcing and project management
- Systems Development Life Cycle (SDLC) issues
- Systems and software testing and acceptance (all levels)
- IT contract intention, meaning and interpretation issues
- Software quality issues and fitness of delivered systems for their intended purpose & environment
- Software requirements elicitation and control
- Valuing IT assets, systems and companies

Mr. Reid has been actively engaged in litigation matters internationally as a consultant and expert in cases involving the failure of large-scale systems and technology related projects/products including those embracing mainframe and multi-tiered computer, internet, enterprise (ERP), point-of-sale, robotics, and e-commerce systems technologies and platforms.

He has testified in U.S. State and Federal Courts and the Court of Federal Claims, and has been engaged as an expert by a "Who's Who" in the world of international business including: the U.S. Department of Justice and President William Clinton; an Asian Stock Exchange; Pepsico; Her Royal Majesty, the Queen of England; Compuserve; Fortune 500 retailers; and ERP software developers --to mention a few.

Mr. Reid graduated with highest honors from Baruch College in New York City and earned his M.S. and M.B.A. degrees from the Wharton Graduate School of Finance in Accounting Theory and Management Information & Control (i.e., Computer Science) respectively. Mr. Reid is a highly published author and has appeared on radio and CNN as an expert in systems technologies.

He is the developer of his four seminal "IT Success Models" which are now part of several University of Southern California (USC) graduate school programs (in software and systems engineering, and in the strategic management of IT) and two law school programs in IT contracting.

