Implementation Success Guaranteed?

What steps need to be taken to assure the outcome of a publishing systems development project?

This document contains abstracted materials from "Software Acquisition Capability Maturity Model (SA-CMM), Version 1.01, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213

Prepared by Niwot Ridge Consulting, 1999

Why are we here?

- *To discover the* Success Factors *for Publishing Systems*
- *To discover the* Attributes *of these success factors.*
- *To discover the* Risk Factors *for various project types.*
- *To understand the steps which lead to success.*
- *To avoid the* Pit Falls *in the process*

What we won't learn here

- *The* magic bullets – *there aren't any, period.*
- *The* turn key *solution to our problem – unless of course you have a* turn key *problem:*
  - This off the cuff comment should actually not be minimized. If there is a problem to be solved which fits (or nearly fits) an existing *Commercial Off The Shelf* product, don't focus on the requirements for the System, adapt the Business Process to fit this System.
  - No matter what anyone says, this is hard work.
  - Specifying the behavior of both the business process and the computing system which supports that business process requires discipline and courage.
  - Controlling the scope of the project requires fortitude.
  - Displacing the current business process with an automated system requires physical as well as mental effort.

Pre–Existing problems with this presentation

- *Scope of the Material*
  - This is a big subject, not amenable to 1½ hours, so we'll move fast.
  - If this were easy, everyone would have flawless Publishing projects and we'd be out golfing instead of having this seminar.
- *Density of the Material (Physical, Logical, Point Sizes)*
  - The slides are intended to be read outside of this room, so they violate all the rules of *good* presentation style:
    - They are dense.
    - They contain (I hope) real ideas and content that you can use later to build your own ideas on.
    - They have references that can be followed to the source of the idea.
- *The Subject Matter Itself*
  - This stuff is hard, somewhat tedious and always looked back upon when the project is in trouble as — *SOMETHING THAT SHOULD HAVE BEEN DONE, BUT OF COURSE WASN'T*.

Attributes of Success

Attributes of Success

- *What components of a project can we actually control?*
  - The scope of work, once the project has been initially defined.
  - The physical and behavioral requirements once the initial requirements have been defined.
- *What components are more difficult to control?*
  - The budget.
  - The time scale.
  - The team members.
- *What are components that actually are out of your control?*
  - The behavior of the system vendor.
  - Global market forces.
  - The ultimate end user's acceptance of the system.

Attributes of Success

- *Habits of a Successful Project Manager* [†]
  - Has a detailed definition of the Critical Success Criteria.
  - Maintains a working knowledge of the schedule and technical details of:
    - The business processes that are being altered by the system
    - The computing system itself
    - The personnel responsible for deploying both the business and technical aspects of the system.
  - Keeps both subordinates and superiors informed of the project's progress
    - Budget tracking for all components of the system. This is a capital project and should be managed like one.
    - *Exit criteria* for each scheduled milestone being met on the schedule

[†] Abstracted from *The 7 Habits of Highly Effective People*, S. R. Covery, Simon & Schuster, 1989.

Attributes of Success

- *Habits of a Successful Project Manager*
  - Maintains incremental success points in the project
    - Demonstratable deliverables which provide everyone in the decision chain with the confidence that *things are going well*.
    - Allows all participants to have *small successes* in order to maintain the momentum.
  - As a Plan B
    - How will the system get the job done when it is not completely deployed or does not function completely?
  - As a Plan C
    - What is the backup plan when the system craters?
    - This plan can also be used as the foundation for the Disaster Recovery Plan.

Attributes of Success

- *The Critical Success Factors Approach* [†]
  - Identify the most critical ingredients which will make the enterprise successful.
  - Define the application systems which will support the critical business functions
  - Analyze, evaluate and justify the proposed application system in light of these critical success factors.

    A Publishing System is core to the business. The question is... *what types of information should be managed and what processes should be altered to support this new paradigm?*

[†] "The Changing Role of the Information Systems Executive: A Critical Success factors Perspective," J. Rockart, *Sloan Management Review*, Volume 24, Number 1, 1982. pp. 3–13.

Attributes of Success

- *Sample Critical Success Factors for the Business Process*
  - Focus the deployment of the Publishing System on measurable business processes.
  - Provide consistent business processes for authoring, reviewing, publishing, storing and delivering documents.
  - Reduce of the cycle time for work activities involving production, handling, distribution and use of paper based documents.
  - Simplify paper based work processes.
  - Reduce or eliminate the cost associated with paper based processes.
  - Improve document security and records management.
  - Improve quality in the production and distribution of documents.
  - Decrease the cost for modifying information prior to publication.
  - Provide direct performance measurements for content creation–centric processes.

Attributes of Success

- *Critical Success Factors for the Technical Infrastructure*
  - An agreed upon networking infrastructure that is capable of addressing the *imaging* load to be placed on it.
  - A standardized database architecture, which includes a Data Base Administrator assigned to the project.
  - A Common Operating Environment (COE) for the desk top.
  - Operational resource assignments to support the system throughout its life cycle.
    - Design analysis
    - Startup
    - Operations

Taxonomy of an Publishing System

Taxonomy of an Publishing System

- *Taxonomy*
  - *The science or technique of classification; the science dealing with the description, identification, naming, classification or organisms; any classification especially the systematic classification or organisms ito hierarchical groups.*
  - Why deal with Publishing Systems in this way?
    - The Publishing market place is populated with products that provide a variety of capabilities, nearly all of which have been developed as a direct result of user needs.
    - In mature products, this creates a plethora of features, which overlap with the needs of the end user community.
    - These feature sets are continually evolving, driven by user needs, market forces, technological fads, competition.
    - One way to sort out this mess is to put the systems into *categories* which define their *behavior* or use rather than their list of features.

Prepared by Niwot Ridge Consulting, 1999

Taxonomy of an Publishing System

- *Three Generic Classes of Publishing Systems*
  - Archival Support
  - Performance Support
  - Authoring Support

Classes of Publishing Systems

- *Archival Support*
  - Provides a repository for *read only* (or nearly read only) information
    - Banking, lease, mortgage service, medical, educational, etc. records
    - Indexing of these records is usually through the original document number, transaction number, account number (static linkage)
  - The archival aspects of the system are the primary *Critical Success Factor*.
    - Optical storage for nearly a *lifetime* (equivalent to microfilm)
    - Legal records management requirements. Microfilm replacement behaviors.
  - Connected to the *Line of Business* support systems, as if the document were in its original form — paper or film.
    - The database record is the primary *document of record*, while the scanned image is the supporting information.
    - Signatures, hand filed fields and other *human* generated information is usually contained in the document.

Classes of Publishing Systems

- *Performance Support*
  - Assisting employees to better execute their tasks.
  - Attributes of a Performance Support Publishing System
    - Focused on User Needs
    - Provides a *Representation* of the user's work process
    - Work processes and business rules guide the user through the daily work activities
    - Provides access to information required to support the decision making process.
  - *Delivers the correct information to the right person at the right place and the right time* [†]

[†] *Electronic Document Management Systems*, L. Bielawski and J. Boyle, Prentice Hall, 1997.

Classes of Publishing Systems

- *Performance Support — System Attributes* [†]
  - Absence of any need for training.
  - Integration of the software in the user's performance.
  - Understanding that *performing*, not *knowing* is the critical success factors.

[†] "Just in Time Knowledge Delivery," K. Cole, O. Fischer and P. Saltzman, *Communications of the ACM*, Volume 40, Number 7, July 1997.
Electronic Performance Support Systems: How and Why to Remake the Workplace through the Strategic Application of Technology, G. J. Gery, Gery Performance Press, Tolland, MA, 1991.

Classes of Publishing Systems

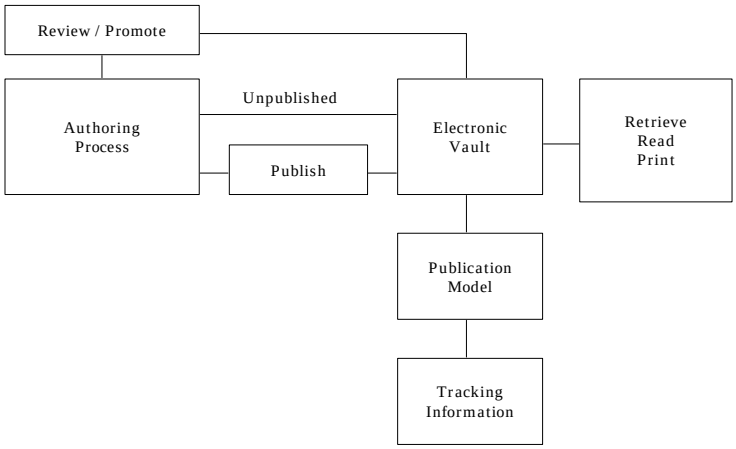
- *Authoring Support*
  - The Publishing system is embedded in an author / review / publish cycle.
    - Publishing formats vary, but are usually in a different form than the original authored text.
      - PDF
      - HTML
  - *Management of Change* is an important attribute to this type of system.
    - Review and update of the various *versions* of a document are managed by the system.
    - Multiple renditions of the document are managed directly by the system.
  - The Web has focused the Publishing systems on these issues
    - How can reliable documents be delivered to HTTP servers?
    - Can HTTP servers be replaced (augmented) by Publishing Systems?
    - *This is the million dollar question asked here*

Components of an Publishing System

- *Layered Components*
  - System Architecture
    - Electronic Vault
    - Middleware
    - Infrastructure
    - Workflow
  - Component Lifecycle
  - System Scalability
  - Interoperability Standards

Components of an Publishing System

- *The Information Management Processes*



Prepared by Niwot Ridge Consulting, 1999

Components of an Publishing System

- *Life Cycle of Various PublishingS Components*

Desktop Applications	These applications provide viewing, printing and markup capabilities for documents. They are usually based on MS Windows and will turn over every 18 to 24 months. Since the underlying operations system is driven by market pressures, the orgnization' ability to control the migration is limited.
Middleware Applications	These applications maintain the data and process models used to describe the documents and their relationships. The underlying technology (RDBMS) is considered stable. Any changes in this infrastructure will take place over 3 to 8 year cycles. The majority of the value of the EDM System is maintained in these applications.
Archive and Storage Applications	These applications provide access to the physically stored documents and their rudimentary indexing information. If the document image format is an industry standard, and it is placed on an industry standard media, the mechanical devices used to access these images changes every 5 to 7 years.

Prepared by Niwot Ridge Consulting, 1999

Getting Started

Strategies are intellectually simple; their execution is not. [†]

[†] "The CEO as Coach: An Interview with Allied Signal's Lawrence A. Bossidy," N. M. Tichy and R. Charan, *Harvard Business Review*, Mar–Apr, 1994, pp. 68–78.

Getting Started

- *A Conceptual Framework...*
  - Provides the *Vision* of the system prior to its actual development.
  - Narrows the scope of the project to those activities which have the highest pay back for the enterprise.
  - Establishes the success criteria for the project prior to defining the details of the software and hardware.
  - Gains consensus from all parties that the project will meet their needs prior to proceeding with development.

Getting Started

- *Road Map for Reengineering the Process*
  - Position for Change
  - Diagnose the Existing Process
  - Redesign the Process
  - Make the Transition
  - Carefully pick the winning applications and the teams that will deploy them:
    - It all has to come together in the end
    - Deploying a Publishing System is no different than designing and building an addition to the physical plant or distribution system.
    - It all has to be managed as a *project* with the proper project controls.

Getting Started

- *Framework for Comparison* [†]
  - Qualitative Information
    - Changes in the organizational structure
    - Changes in work flows and functions performed
    - Interface changes
    - Changes in technology
    - Impact of the Publishing on the organization
  - Quantitative Information
    - Comparison of data flow diagrams
    - Comparison of resources required
    - Investment analysis

[†] "A Reengineering Framework of Evaluating a Financial Imaging System," H. C. Lucas, D. J. Berndt and G. Truman, *Communications of the ACM*, Volume 39, Number 5, pp. 86–96, May, 1996.

Getting Started

- *Some target improvements*
  - Eliminate manual document entry and recording processes, through scanning and indexing at time of document creation.
  - Provide instant access to information, at the *point of need*.
  - Allow bidirectional transfer of documents traditionally handled through hardcopy to and from the document repository.
  - Provide indexes that model the document's usage, not necessarily the document's *Title*.

Getting Started

- *Key Factors to Success*
  - Project Sponsors
  - IT Support
  - Seamless Integration
  - Staged Development and Deployment
  - Technology Review and Readiness
  - Network Infrastructure
  - *Additional* Factors

Getting Started

- *Obstacles to Success*
  - Lack of cross functional project team
  - Lack of executive consensus
  - Lack of senior management champions
  - Limitations of existing systems
  - Resistance to change
  - Unrealistic expectations

Getting Started

- *Cost Justification Approach*
  - Define current expenses for each department
  - Define potential savings by using Publishing
  - *Book* the difference as contribution to Publishing funding.
  - Analysis Process:
    - Define Data Entities
    - Define Process Flow
    - Assign Cycle and Process Times and Costs to each process step to build the *as is* cost model.

Getting Started

- *Issues with the* Traditional *Approach...*
  - Automating the *current process* produces limited benefits over the long run.
    - Although throwing out the existing system is not the first choice either — avoid drastic measures.
  - Process analysis oriented toward *department* functionality *rather than* process *functionality* misses the point...
    - the process functionality is independent of the department organization.
    - the process is *making product* and the supporting functions, not necessarily the current department organization.
    - Publishing augments the business, it is not *the business*.

Getting Started

- *An* Alternative *Approach…*
  - Define the processes as they *should be*.
  - Deploy the Publishing System in the following order:
    - Electronic Vault (EV) — which serves as the basis for all follow–on activities.
    - Re–define processes to take advantage of the EV.
    - Avoid specific solutions on the desktop, until the EV is well defined.
    - For the moment, place the *authoring* process outside the Publishing System, so the EV activities become the core of the system, rather than the document creation process.

Getting Started

- *Sponsorship – Key to Success*
  - *The Sponsor is the person(s) who:*
    - has the authority to commit all resources necessary to implement the system and the changes associated with it.
    - decides what changes will be implemented and when.
    - commits the necessary resources to implement the change.

Getting Started

- *What can be done when buying COTS?* [†]
  - *Identify* – search for and locate risks before they become problems.
  - *Analyze* – process risk data into decision–making information.
  - *Plan* – translate risk information into decisions.
  - *Track* – monitor risk indicators and mitigate actions.
  - *Control* – correct deviations from planned risk actions.
  - *Communicate* – provide information and feedback internal and external to the project.

[†] *Rightsizing the New Enterprise: The Proof, Not the Hype*, H. Kern and R. Johnson, Prentice Hall, 1996.

Getting Started

- *Don't Underestimate the Management Effort*
  - A typical project will have 100 to 300 identifiable tasks. Don't micromanage these tasks, but don't overlook the details. This is no different than any other project management activity.
  - Remember – ya pay's me now or pay's me later….

Methodologies of Project Management

It is the essence of a methodology – as opposed to a method, or technique – is that it offers a set of guidelines or principals which in any specific instance can be tailored both to the characteristics of the situation in which it is to be applied and to the people using the approach. [†]

[†] *Issues in Requirements Elicitation*, M. G. Christel and K. C Kang, CMU/SEI-92-TR-12, Software Engineering Institute, September, 1992.

Prepared by Niwot Ridge Consulting, 1999

Methodologies of Project Management

- *Motivations for a Methodology*
  - Requirements analysis is a process in which *what is to be done* is elicited and modeled.
  - This process must deal with different view points, since multiple view points are concurrently in effect when the system is in operation.
    - Users
    - Administrations
    - Financial payback streams
  - The product of the requirements analysis is a document which contains the *requirements* for the operational system
    - This may appear obvious and redundant, but its not.
    - This requirements document is not a list of features for the Publishing System.
    - It describes the system behavior independent of the feature set.
    - The features needed to implement the requirements only comes *after* the requirements have been defined.

Methodologies of Project Management

- *Deployment of a* Localized *Methodology* [†]
  - Software applications model certain aspects of the real world within some larger real world model.
    - The software system is simply an abstraction of the what actually takes place during any specific business operation.
  - Critical design activity involves establishing that external behavioral model, not a computational model that executes it.
    - This point is made again to separate listing Publishing System features and confusing them with defining requirements.
    - It is not the software itself or the underlying hardware system that is most interesting to the end user (or buyer). It is the behavioral aspects of the system that increase the business benefits to having deployed the system in the first place.

[†] *Beyond Programming: To A New Era of Design*, B. Blum, Oxford University Press, 1995.

System Requirements Analysis

- *What are Requirements?* [†]
  - Requirements are defined in the early stages of a system's development as a specification of what should be implemented.
  - They describe *HOW* the system should behave.
  - They describe the *Properties* or *Attributes* of the system.
  - They may place constraints on the system.
  - Requirements describe
    - A user facility
    - A general system property
    - A specific constraint

[†] *Requirements Engineering: A Good Practice Guide*, I. Sommerville and P. Sawyer, John Wiley & Sons, 1997.

Systems Requirements Analysis

- *What is Requirements Engineering?*
  - Requirements Engineering is a structured process which is used to derive, validate and maintain a Systems Requirements Analysis (SRA).
  - A complete requirements description should include:
    - What activities are to be carried out.
    - The structure and scheduling of these activities
    - Who is responsible for each activity
    - The Inputs and Outputs to / from the activity
  - The primary activity of Requirements Engineering is eliciting the requirements from various points of view so:
    - All parties concur that the system behavior is acceptable.
    - The system meets all the financial, performance, reliability, legal and regulatory requirements.

Requirements Engineering

- *Requirements Elicitation*
  - *Assess the system feasibility*
    - ✓ Before incurring any expenses in discovering the requirements for the system, some form of feasibility needs to be carried out.
    - ✓ This study can determine if the system can be implemented given the existing technology, personnel and politics.
    - ✓ The feasibility study will also confirm that the system can provide adequate returns for the investment, in a time frame acceptable to the user.
  - *Understand the organizational and political aspects of the project*.
    - ✓ Who are the stakeholders, what are their hidden agendas, who will actually make the system work at the end of the day, how will the system be measured, who will sign off that the system actually works?
  - *Identify and integrate the stakeholders*.
    - ✓ Along the same lines as above, the success criteria is held by the people doing the work and writing the checks.
    - ✓ They must be involved in all phases of the project.

Requirements Engineering

- *Requirements Elicitation*
  - *Define the system's environment*.
    - The understanding of the operating environment is key to the technical success of the project.
  - *Use business drivers to define the system*.
    - Not simply the availability of technology. Why solve a problem with technology that doesn't have any pay back.
    - The *No value added processes* should be eliminated (except maybe in the regulatory world).
    - The understanding of the payback for every process is *mandatory* if the system is ever to be successful in the eyes of the people who paid for it.
    - The system must of course work for the end user, but that will become irrelevant if it doesn't meet the financial goals.

  This is one key to success – know the business reasons for doing something. The technical issues can always be sorted out later.

Requirements Engineering

- *Requirements Elicitation*
  - *Define the constraints on the Domain*.
    - By defining the constraints imposed by the external and internal legacy systems, business processes and other intangible processes, their impact on the outcome of the current system can be determined.
    - One approach used is called “Impact Analysis” in which each processes is assessed for its impact on other processes.
      - Who does what to whom?
      - If a process is changed, what other processes are impacted?
      - What is the cost / benefit of all these impacts?
    - The impact (positive or negative) is documented and mitigating processes are defined unfavorable impacts.

Requirements Engineering

- *Requirements Elicitation*
  - *Use a multilevel attribute based requirements analysis approach*.
    - There are no simple solutions to any of the problems encountered in the Publishing world.
    - Some form of weighted analysis needs to be used.
    - There are many different influences in the requirements for a typical Publishing system.
    - These influences involve end users, managers, regulatory agencies and other organizations whose work is directly or indirectly affected by the system.
    - All of these and more are potential sources of requirements for the system. This approach results in a “Viewpoint” based description of the requirements.
    - The many *Viewpoints* of the system must be addressed before the requirements can be clearly understood. (The “let’s make a list” approach falls woefully short here).

Requirements Engineering

- *Requirements Elicitation*
  - *Prototype the poorly understood requirements*.
    - By building prototypes (not pilots) the requirements for the system can be "self developed".
    - These prototypes can be developed in many ways:
      - Rapid Development Tools
      - Screen Cams
      - Static Pictures
      - Chalk Talks
      - Hand Waving
    - The prototype stimulates discussion of the behavior of the system in a way no textual description can. End Users can experiment with the prototype to refine their ideas about the system requirements.
  - *Use Scenarios to Elicit Requirements*.
    - In the Object world these are called Use Cases.
    - Use Cases are NOT lists of features, but are behavioral descriptions of the system for a specific task or function. "If I want to do X, push button Y and read the result Z on screen 123".

Requirements Engineering

- *Requirements Elicitation*
  - *Define the Operational Processes*.
    - The object of the system is to support a business function.
    - These business processes are part of the system analysis and must be as well understood as the technical aspects of the system.
    - These processes are both data and procedural.
      - A definition of the data model is not sufficient.
      - How the data is used, by whom and for what purpose is also needed.
    - By developing these models the system can be understood from many aspects, including, performance, reliability, (and all the other ..abilities) and most importantly from the financial point of view – when will the user make money with the system?
  - *Reuse and Exit Strategy Requirements*.
    - The system will age and become obsolete at some point.
    - How can the components be reused or decommissioned with the minimum of impact on future users?
    - These figures need to be included in the IRR or ROI calculations.

Requirements Engineering

- *Problems with Requirements Specification [†]*
  - A prototype is worth 100,000 words – producing a working prototype that can be used to *demonstrate* a concept is much better than writing about it.
  - Gold Plating the Requirements – fixing the requirements too early in the process tends to have the requester add all kinds of unnecessary feature, just to cover the bases.
  - Inflexible Point Solutions – optimizing the requirements around a small set of *needs* restricts the solution space.
  - High Risk Downstream Capabilities – the desire to *get something out* leads to short cuts that may never catch up with the real requirements.
  - Off Target Initial Release – *Let's find out what the users need and build an initial version they can improve upon*, leads to a first installation that does not meet anyone's needs.

[†] "Anchoring the Software Process," B. Boehm, *IEEE Software*, Volume 13, Number 4, July 1996, pp. 73–82.

Attributes of the Software Acquisition Process

Attributes of the Software Acquisition Process

- *A Formal Methodology*
  - Buying software based systems has been around as long as there has been software system salesmen (and sales women).
  - There is a methodology in the market place that can help you the various steps evolved in procuring software, integrating that software and move that procured software in production.
  - *Software Acquisition Capability Maturity Model (SA-CMM), Version 1.01* is the document describes a framework for improving the process of acquiring commercial off the shelf software. CMU/SEI-96-TR-002, ESC-TR-96-020, December, available at ftp://sei.cmu.edu/
  - The next few slides describe the SA-CMM process and its applicability to the problem at hand.
  - The Copyright notice at the beginning of this presentation covers the work that follows.
  - *THE INFORMATION PROVIDED BELOW IS A HIGHLY ABSTRACTED VERSION OF THE ACTUAL METHODOLOGY. NO INTENT IS MADE TO CONVEY THE MATERIAL HERE AS THE COMPLETE METHODOLGY.*

Attributes of the Software Acquisition Process

- *Architecture of the SA-CMM*
  - Defines Five (5) levels of maturity. Each level indicates the capability of the *acquiring* organization to deal with the issues associated with purchasing, installing and transactioning to production a Commercial Off The Shelf (COTS) software system.
  - Key performance areas are defined for each level as well as the attributes that indicate whether the implementation of the key process areas can be effective, repeatable and lasting.
  - The five common features are:
    - Commitment to perform
    - Ability to perform
    - Activities performed
    - Measurements and Analysis
    - Verifying Implementation

Attributes of the Software Acquisition Process

- *Level 2 - The Repeatable Level*
  - This discussion assumes that you will be buying major software components from software vendors.
    - These components may be integrated into the final system.
    - These components may operate as a *turn key* system
    - In all cases it is assumed that the system is based on a COTS product
  - This should be the minimum level at which the acquiring organization operates.
  - Key Process Areas:
    - Software Acquisition Planning
    - Solicitation
    - Requirements Development and Management
    - Project Management
    - Contract Tracking and Oversight
    - Evaluation
    - Transition to Support

Attributes of the Software Acquisition Process

- *Requirements Development and Management*
  - Goals
    - Software related contractual requirements are developed and maintained in conjunction with the end user and other affected groups.
    - Software related contractual requirements are unambiguous, traceable and verifiable.
    - The software related contractual requirements baseline is established and managed.
  - Activities Performed
    - The project team performs its activities in accordance with its documented requirements development and management plans.
    - The project team develops and baselines the software related contractual requirements and places them under change control early in the project, but not later than release of the solicitation package.
    - The project team appraises system requirements change requests for their impact on the software being acquired.
    - The project team appraises all changes to the software related contractual requirements for their impact on performance, architecture, supportability, system resources utilization and contract schedule and cost.

Attributes of the Software Acquisition Process

- *Project Management*
  - Goals
    - Project Management activities provide effective management control of the software acquisition project.
    - The performance, cost and schedule objectives of the software acquisition project are defined, measured and controlled throughout the software acquisition.
    - Problems discovered during the software acquisition are managed and controlled.
  - Activities Performed
    - The Project Team performs its activities in accordance with its documented software acquisition management plans
    - The organization of the project provides for the management of all project functions
    - The software acquisition management activities of the project team are directed to accomplish the projects objectives
    - The software acquisition management activities of the project team are controlled
    - The project team implements a corrective action system for the identification, recording, tracking and correction of problems discovered during the acquisition process.

Attributes of the Software Acquisition Process

- *What Does All This Mean, and Why Do We Care?*
  - This is one methodology.
  - Many exist
  - Pick One
  - The process of identifying a need, acquiring a system to meet that need and transitioning that system to production is a *Generic* process.
    - It requires skilled project management personnel
    - It requires a clearly defined and measurable set of requirements.
    - It requires the same level of commitment that any capital project does.
  - *There is no magic bullet here, just hard work, clear thinking and a dedicated focus on producing an outcome that can be measured for success.*

Managing the Risks

Managing the Risks Associated with the Project

- *Risk Assessment Approach*
  - Types of Risk
  - Severity of Each Risk Type
  - Risk versus the various stages of the project
  - The Project Management process usually does not consider risk in any financially meaningful way.
    - Risk is the probability of a specific undesirable event occurring during a specified time interval.
    - One typical undesirable event is a negative ROI for the project.

Managing the Risks Associated with the Project

- *The Nine Principals of Managing Risk* [†]
  - Shared Vision
  - Forward Looking Search for Uncertainties
  - Open Communications
  - Value of Individual Perception
  - Systems Perspective
  - Integration into the normal project management
  - Proactive Strategies
  - Systematic and Adaptable Methodology
  - Routine and Continuous Processes

[†] *An Introduction to Team Risk Management*, R. P. Higuera, et al, CMU/SEI-94-SR-1, Software Engineering Institute, May 1994.

Managing the Risks

- *Designing an Publishing System is different* [†]
  - The answer to systems analysis problem is abstract and not realized in a way that is easy to comprehend by the users of the Publishing System.
  - The Publishing System must conform to a complex, non–specific set of human behaviors as well as a set of explicit data and programmatic interfaces.
  - Since the resulting system is realized in software, some form of *correctness* is required for the business to operate without error.

[†] "Understanding the Elements of System Design," in *Critical Issues in Information Systems Research*, edited by R. J. Boland and R. A. Hirscheim, John Wiley & Sons, 1987, pp. 97–111.

Managing the Risks

- Breakdown *in the Analysis Process is...* [†]
  - The use of ineffective design activities
  - The undesirable consequences of these ineffective activities
  - The production of activities that are difficult to perform because they tax the designers' cognitive resources
  - The failure to identify the causes of these ineffective and difficult design activities.

    These breakdowns are necessary to move the analysis process forward [‡]

[†] "Communication Breakdowns and Boundary Spanning Activities in Large Programming Projects," H. Krasner, B. Curtis and N. Iscoe, in *Empirical Studies of Programmers: Second Workshop*, Abex Publishing Corporation, pp. 47–64.

[‡] "A Methodology for Studying Software Design Teams: An Investigation of Conflict behaviors in the Requirements Definition Phase," D. B. Walz, J. J. Elam, H. Krasner and B. Curtis, in *Empirical Studies of Programmers: Second Workshop*, Abex Publishing Corporation, pp. 83–99.

Managing the Risks

- *Taxonomy of* Breakdowns
  - Structural breakdown
    - Lack of design schema
    - Poor prioritization of the issues
    - Poor acquisition, sharing and integration of domain knowledge
    - Differing mental models among team members
  - Short Term memory Limits
    - Difficulty conceptualizing the constraints
    - Difficulty performing mental simulations
    - Difficulty keeping track of sub–problems
    - Difficulty scaling up partial solutions

Managing the Risks

- *Major Problems in Systems Analysis* [†]
  - Thin application domain knowledge
  - Fluctuating and conflicting requirements and a need to maintain multiple models of the system
  - Communication and coordination breakdowns

[†] "A Field Study of the Software Design Process for Large Systems," B. Curtis, H. Krasner and N. Iscoe, *Communications of the ACM*, 31(11), November 1988, pp. 1268–1287.

Mitigating the Risks

- *Planning the System Deployment*
  - Develop a hardware plan (storage, memory)
  - Develop a contingency plan
  - Develop a security plan
  - Establish a Change Management Strategy
  - Establish a data migration strategy
  - Manage the system component life cycles
  - *Good project managers must re-plan the project as it unfolds.*

Mitigating the Risks

- *Life Cycle Plan* [†]
  - Having a plan of how to mitigate risks is as important as actually mitigating the risks.
  - Using the Way, What, When, Who, How, How Much Methodology to stay out of trouble
    - Objectives: *Why* is the system being developed?
    - Milestones and Schedule: *What* will be done by *When*
    - Responsibilities: *Who* is responsible for a function? *Where* are they organizationally located?
    - Approach: *How* will the job be done, technically and managerially?
    - Resources: *How Much* of each resource is needed?

[†] "Anchoring the Software Process," B. Boehm, *IEEE Software*, Volume 13, Number 4, July 1996, pp. 73–82.

Avoiding Pitfalls

Avoiding Pit Falls

- *Types of Pit Falls* [†]
  - Conceptual Pit Falls
  - Political Pit Falls
  - Management Pit Falls
  - Analysis and Design Pit Falls
  - Implementation Pit Falls
  - Quality Assurance Pit Falls

[†] Abstracted and reformed from *Pitfalls of Object–Oriented Development: Symptoms, Consequences, Detection, Extraction and Prevention*, B. F. Webster, M&T Books, 1995.

Avoiding Pit Falls

- *Conceptual Pit Falls*
  - Deploying Publishing for the wrong reasons
    - There are many valid reasons for deploying an Publishing System. However they usually have nothing to do with the availability of the technology.
  - Assuming a *Software Product* can solve your problem
    - It can't until the problem is clearly defined and the alterative *proven* to make financial sense.
  - Assuming Publishing is a mature software paradigm
    - It's getting there, but Publishing is still on the curve to maturity
  - Confusing Products with Solutions
    - Solutions are business process based, products enable the solution
  - Confusing Training with Skill
    - Publishing Systems are not *Plug and Play*.
    - Reading books and attending seminars and trade shows is a beginning, but true skill comes from experience.
  - Confusing Prototypes with Mature Technology
    - Trade show demos hardly ever work in production environments

Avoiding Pit Falls

- *Political Pit Falls*
  - Not educating and eliciting management support in the beginning
    - Eventually some has to pay. Knowing how much and why on day one is better.
  - Underestimating the organizations resistance to change
    - No matter what the direct benefits, the Publishing System usually impacts the work flow processes
  - Overselling the capabilities of Publishing
    - It's really just a simple matter of controlling the documents with machines. Publishing does not replace good business management.
  - Becoming too vendor focused (Theological wars can be bloody)
    - Vendors come vendors go, your business (hopefully) remains
  - Not recognizing that system architecture can be political too
    - Think about the system from the view *inside* the glass walled room.
  - Becoming over zealously about product features
    - Feature–itis creates confusion about how the system will earn its keep in the eyes of the buyer.

Avoiding Pit Falls

- *Management Pit Falls*
  - Adopting Publishing without clear and measurable objectives
    - No measurable goals, not measurable success
  - Abandoning the *Best Practices* of today for a technology fix
    - The business must keep running
  - Attempting too much, too soon, too fast, with too little experience and resources.
    - Pioneers have arrows in their backs, know where you're going and why.
  - Assuming a linear deployment strategy
    - Systems is this type almost never scale up in a linear fashion
  - Allowing *Scope Creep*
    - Don't allow scope creep, period
  - Mistaking prototyping or demonstration software with production deployment.
    - Seeing trade show demos is not the same as running it in production
  - Not managing the risks
    - They're always there
  - Lying to your self about the project success possibilities
    - All's well that Only ends well is usually not true. You may become road kill along the way.

Avoiding Pit Falls

- *Analysis and Design Pit Falls*
  - Underestimating the need for clear requirements
    - Without measurable goals, the success criteria can not be determined
  - Underestimating the difficulty of analysis and design
    - When done properly, analysis and design are difficult
  - New Wine in Old Bottles
    - New paradigms into old business systems spoils both
  - Not looking out for blind spots
    - This business of fraught with risks, *be careful out there*.
  - Building too general purpose of a solution
    - In an effort to deploy a general system, the constraints of the original problem can be overlooked
  - Making things too complex
    - Undue complexity is a danger single in the Publishing business
  - Design by enumeration
    - Listing all possible situations and processes that the system must address will lead to overlooked details. Designing by *abstraction* allows downstream flexibility.

Avoiding Pit Falls

- *Implementation Pit Falls*
  - Making too many promises without enough time, money and technology to keep them.
    - Remember once again that what you see in the literature, demonstrations, trade shows is usually not something you can take to the production environment.
  - Leaving the details for later
    - *The Devil is in the Details*
  - Redeploying subsystems in a single bound
    - Once a subsystem (e.g. scanner too slow) is discovered not to meet the requirements, rapidly redeploying something else can lead to the next redeployment. Think carefully about each step.
  - Failing to document the key concepts and decisions
    - Why are you doing this?
    - What was the original goal here?
    - Have you lost sight of the well defined and measurable success criteria.
  - Being seduced by the Dark Side
    - Following all of the guidelines by no mans assures success.

Avoiding Pit Falls

- *Quality Assurance Pit Falls*
  - Forgetting the combinatorial issue
    - There are lots of moving parts in the system, not all of which work as specified.
  - Neglecting the testing of individual subsystems
    - Almost all vendors buy *some* of their pieces from others. Testing in your environment is vital to the integration process.
  - Thinking about a Test Plan after the fact
    - This is a computer software system being installed and integrated on site, have a honest test plan.
  - Underestimating the support effort
    - Getting the system into production is just as difficult as all the activities performed prior to the production roll out

Avoiding Pit Falls

- *The Classic Mistakes*
  - Undermined Motivation
  - Uncontrolled Problem Employees
  - Abandoning the Planning Process Under Pressure
  - Shortchanging the Upstream Activities
  - Shortchanging the Quality Control Activities
  - Lack of Control Over Feature Creep
  - Looking for the *Silver Bullet*
  - Wasting Time on the Perfect User Interface
  - Insufficient User Input
  - Overly Aggressive Schedules
  - Adding Staff at the Last Minute

Wrap Up

Wrap Up

- *What Does All This Mean?*
  - Managing an Publishing System Project is no different than managing any other complex IT project. In addition though:
    - The objects being manipulated are large and cumbersome
    - They are used in the daily operation of the plant or business
    - The plant or business cannot stop while the new system is installed
  - The vendors of products that can address the needs want you to believe...
    - …there are *out of the box* solutions – they are but they are rare.
    - …there are ways to *shorten* the process of defining the need, selecting the appropriate components, assembling the system and getting it installed and operational – there aren't.
  - The Publishing System is a *Line of Business* application in the same way Accounts Payable or Payroll is – if the system is to be successful the business will depend on it for its proper operation.

Wrap Up

- *What Should be Done About the Issues Raised Here?*
  - Look at the systems being presented at the show in light of the following:
    - Is there a clear statement of the business problem and the payback model?
    - Are there components (software subsystems) that can be assembled into a complete system to meet these identified needs?
    - If so, when will the investment be earned back?
    - What business processes must be changed to adapt to the offerings being presented at the show?
    - Can this be done in your organization?
  - Build a project plan that is realistic
    - Requirements
    - Detailed design
    - Installation, Training, Startup and Operational Support
    - Measured payback analysis to confirm the validity of the project