Communications of the Association for Information Systems

Volume 2 Article 17

September 1999

Software Project Management: The Manager's View

Jaak Jurison
Fordham University, jurison@fordham.edu


Communications of the Association for Information Systems

Volume 2, Article 17
September 1999

SOFTWARE PROJECT MANAGEMENT:

THE MANAGER'S VIEW

Jaak Jurison
Graduate School of Business
Fordham University
jurison@mary.fordham.edu

TUTORIAL

SOFTWARE PROJECT MANAGEMENT:

THE MANAGER'S VIEW

Jaak Jurison
Graduate School of Business
Fordham University
jurison@mary.fordham.edu

ABSTRACT

As businesses become more dependent on information technology for their operations, IS managers are under increasing pressure to deliver quality applications software on time and within budget. Thus, in addition to their technical skills, they must master the necessary management skills to lead and control software development projects.

The purpose of this tutorial is to present the fundamental concepts of modern project management and show how these concepts can be applied to software development projects. The tutorial presents a broad overview of current software project management practices that evolved over the years from a variety of complex projects. The subject is presented from the manager's rather than from the developer's perspective. The focus is on large and complex projects because these projects are the most challenging and in need of an effective project management discipline.

Keywords: Project management, software development, project planning, project control, team building, critical success factors.

I. INTRODUCTION

Projects are not new. People have been working on projects since the early days of organized work. The Egyptian pyramids, the Greek Parthenon, and the Great Chinese Wall are examples of major projects of historic importance. What is new is the way we manage projects. Project management as a special form of management evolved from the work done on large-scale military projects where an organized approach was necessary to manage the complex interrelationships among an enormous number of different tasks performed by many different specialists. In recent years project management emerged as a major new form of management to deal with the complexities of knowledge-based teamwork in organizations facing rapidly changing business environments. Project management provides managers with powerful methods and tools for planning, organizing, and managing team-based activities for accomplishing specific objectives.

No other management activity can benefit more from effective project management than software development. Practically all software development efforts are undertaken as projects. These projects are generally complex and their development takes place in a dynamic environment where business conditions and technologies change during the project. Users are often unsure of their needs and frequently change requirements midway through the project. As a result, the software industry is plagued by cost overruns, late deliveries, poor reliability, and user dissatisfaction (Abel-Hamid and Madnick, 1991).

Why is managing projects so difficult? Why are we seeing so many project failures, especially in software development? Some of the difficulties stem from the inherent nature of the product, others are management related. Among the common software related problems are:

- *Intangibility*. Software, unlike hardware, is intangible. As a result, software is difficult to manage because it contains no visible milestones to measure progress and quality.
- *Complexity*. The sheer complexity of software makes it difficult for people to comprehend it, creating not only technical, but management problems as well.
- *Volatility of requirements*. Software requirements are under constant pressure for change. Because software can be changed more easily than hardware, change is a way of life in software development.

Among the management-related difficulties the following are most frequently cited in the project management literature:

- Poorly defined goals and specifications
- Lack of project plan
- Unrealistic deadlines and budgets

Although some projects fail for technical reasons, most project failures are caused by people who ignore the principles of good project management. The purpose of this tutorial is to present these principles and show how they can be applied to the development of information systems.

The paper is organized as follows. Section II presents an introduction to project management, defines the key dimensions of project management, and describes the project life cycle. The process of project planning is described in Section III. It covers project definition, cost and schedule estimating, and risk assessment. Section IV addresses the issues of project organization. It describes the process of setting up the project organization: how to select the project manager and team members, and how to structure the team. The following section deals with the issues of project control and evaluation. Section VI addresses team building and project leadership issues. Critical success factors

for software project managers are presented in Section VII, with concluding remarks in Section VIII. Two appendices provide information on earned value calculations and project management software packages.

Project management is a broad subject that cannot be described completely in a single paper. Therefore a bibliography is included for those who want to dig deeper into any of the topics discussed in this tutorial.

II. PROJECT MANAGEMENT

THE NATURE OF PROJECTS

A project is a temporary assemblage of resources to solve a one-of-a-kind problem. Projects come in various sizes and types. They range from small projects like developing a spreadsheet-based sales plan to large enterprise-wide projects employing hundreds of people working for several years. But regardless of the size, all projects exhibit common characteristics that distinguish them from other types of work.

- Projects have specific objectives.
- Projects must be completed within a specific time period. They have well defined beginnings and ends.
- Projects must be completed within a given budget. Although some projects may have loosely defined budgets, all projects have budgetary constraints.
- Projects are carried out by teams. The assignment of people to a project team can be full-time and/or part-time, depending on the specific needs of the project.
- Projects are unique. While the degree of uniqueness may vary from project to project, all projects are essentially one-of-a-kind, nonrecurring undertakings.

KEY DIMENSIONS OF PROJECT MANAGEMENT

Project management is a series of activities associated with carrying out the project as effectively as possible. Kerzner (1989, p.4) defines project management as "the planning, organizing, directing, and controlling of company resources for a relatively short-term objective that has been established to complete specific goals and objectives." The purpose of project management is to provide focus for using the resources to achieve a specific objective. In short, the fundamental objective of project management is to "get the job done," to reach the objectives within

- time,
- cost, and
- performance.

These three variables are the critical project dimensions which require continual project management attention.

Time refers to the timeliness of progress relative to the schedule. The key questions to be addressed are: "Is the project on schedule?" or "How large is the schedule slip?" Cost means the expenditures for project resources, usually measured in terms of expenditure rate and cumulative expenditures. Performance is the degree to which the objectives or specifications are met. In information systems projects, performance is specified in terms of certain functional and quality requirements, some of which are quantitative, some qualitative.

These three dimensions provide the focal point for all project management efforts. They require the project manager’s undivided attention and energy. They are also the constraints within which project management operates. Therefore they are sometimes referred to as a triple constraint. The challenge of project management lies in finding a balance among these constraints.

More recently, managers added a fourth constraint: good client relations. The ultimate measure of project success is the client. If in the process of meeting the three critical dimensions the manager or the project staff alienate the client, the project fails. A project can be considered a success only if the client, whether it is a group of internal users or a client in another company, is satisfied with the results.

Client interaction is particularly important for information systems (IS) projects. As an increasing number of new IS projects become more strategic and involve business process reengineering, management of organizational change is an integral part of project management.

In addition to the traditional approach to project management that is focused on controlling cost, schedule, and technical performance, this paper highlights the importance of two factors in software project management: *visibility* and *commitment*.

Software is mostly invisible and software projects also tend to be invisible. To be successful, project managers must make the product (the software being developed) and the project (the development process) visible. Project goals, system requirements, project plans, project risks, individual responsibilities, and project status must be visible and understood by all parties involved. Only then can the project team make informed decisions and have a reasonably good opportunity for success.

Commitment of resources and support is needed from the sponsoring organization. Furthermore, all project team members must be committed to the over-all objectives of the project and to their assigned tasks and responsibilities.

PROJECT LIFE CYCLE

All projects follow a series of phases as they progress from start to completion. The phases are characterized by the types of tasks to be performed

and decisions to be made. This series of phases is referred to as the project's life cycle. A clear understanding of these phases helps project managers to organize the work and to allocate and control resources for the achievement of the goals.

While project life cycle can be defined in many different ways, all projects can be broadly broken into four generic phases:

- Project conception
- Planning
- Execution
- Termination

Table 1 shows the four generic phases and the appropriate management actions for each.

Table 1. Project Phases

Conceptual	Planning	Execution	Termination
Identify needs	Prepare plans	Perform work	Transfer responsibility
Establish goals	Develop budget	Procure material	Release resources
Determine feasibility	Develop schedule	Build and test	Transfer team members
Prepare proposal	Assemble project team	Verify performance	Reward people
Estimate time and resources (rough)	Build and test prototypes	Modify as required	Conduct review
Identify key people	Get approval for next phase		
Get approval			

The fundamental purpose of the conceptual phase is to determine the feasibility of the project. The objectives are examined in the context of the business environment, alternatives are defined and evaluated, and preliminary estimates of cost, schedule and risk are made. This phase culminates in a

decision whether to go ahead with the project or not. Many projects do not advance beyond this stage - they may turn out to be technically impractical, too risky, or their projected costs may outweigh their benefits.

In the planning phase (sometimes referred to as the definition phase) the performance, cost, and schedule estimates are refined to a point where detailed plans for project execution can be made. Budgets and schedules are developed, the project team is formed, and a project management system is established to guide the management of the project.

The execution phase entails carrying out the work as defined in the planning phase. In this phase, the program manager's responsibility is to manage the resources necessary to accomplish the objectives. The emphasis of responsibilities shifts from planning to control. For IS projects, the execution phase frequently extends beyond delivery of the end product and includes system implementation, the process of putting the system into operation in the client’s organization. It is not uncommon to have system implementation handled by a separate project team because the implementation team often must function as a change agent rather than as a developer. System implementation introduces a new set of project management challenges that are beyond the scope of this tutorial.

The termination (or divestment) phase is the phase in which the project activities are phased out. It can be triggered either by premature termination or by successful achievement of the goals. In either case, certain activities are necessary to wrap up the project.

The level of resources consumed during a project varies from one phase to the next. Typically, resources build up gradually during the initial phases, then peak in the execution phase, and drop off in the termination phase. Figure 1 shows the resource needs of a typical project, expressed in terms of staff-hours

over the life cycle of the project. While the timing of the peak and the overall level of resources may vary from one project to another, the general shape of the curve tends to be fairly consistent across all large-scale projects.

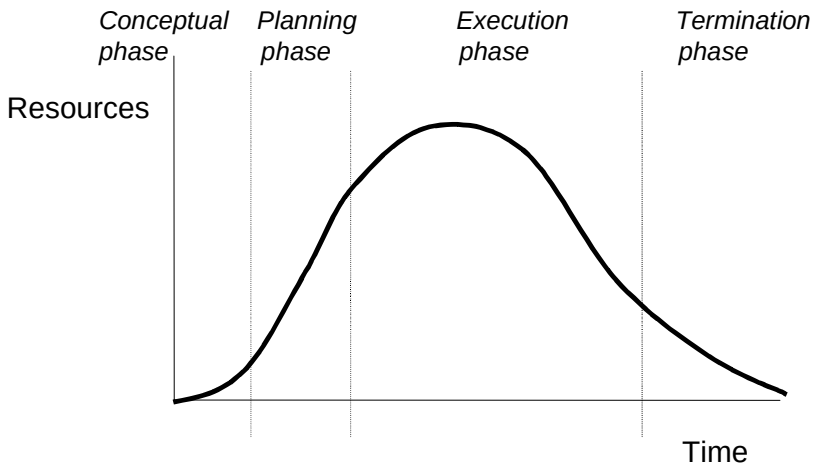

Figure 1. Project Resource Utilization Curve

III. PROJECT PLANNING

The foundation for successful software development project is strong up-front planning. Many project failures can be traced to poor planning. Unrealistic deadlines and budgets, poorly defined goals and objectives, and a lack of project plan are the most frequently cited causes for project failure.

Project planning consists of determining what activities and what resources are needed and how they are to be managed to ensure successful completion of the project. Planning for a software development project includes the following activities:

- Project definition
    1. Defining objectives and requirements

2. Choosing a development process
3. Defining the work

- Estimating
  1. Determining the size of the product
  2. Scheduling
  3. Cost estimating and budgeting
- Risk assessment

PROJECT DEFINITION

Defining Objectives and Requirements

The planning cycle starts with firming up the goals and objectives and determining the requirements for the system. The objectives provide over-all direction for the project and help define the deliverables. A deliverable is a tangible result delivered upon completion of a task. It may be in many different forms, such as a computer file, a report, a manual, or installed hardware. Clear and unambiguous definition of all deliverables is essential. Technical requirements should be defined early. In many cases it may be necessary to build and test a prototype to develop a good understanding of the system's needs and requirements. A prototype is particularly useful in situations where the client is unsure about the requirements.

A clearly defined requirements specification, agreed upon by both the client and the development team, ensures that the client's needs are correctly understood before starting design work. The requirements document is, in effect, a contract between the client and the development team. It specifies what the product must do, but not how. It serves as a guide for design activities and as the baseline for controlling any technical changes that may be needed during the project.

Choosing the Development Process

The choice of the software development process has a significant influence on the project's success. The appropriate process can lead to faster completion, reduced cost, improved quality, and lower risk. The wrong process can lead to duplicated work efforts and schedule slips, and create continual management problems.

Software process models, with some variations, all evolved from the classical waterfall model, developed in the late 1960s and early 1970s. It is still the best and most widely used framework for describing the software development process. It reduces the complexity of the development process by breaking a project into a series of basic steps. The termination point for each step is clearly defined by a distinct set of deliverables, e.g., a requirements specification or coded and tested software modules. Each successive phase can be started only after the preceding phase is complete. A simplified version of this model is shown in Figure 2.

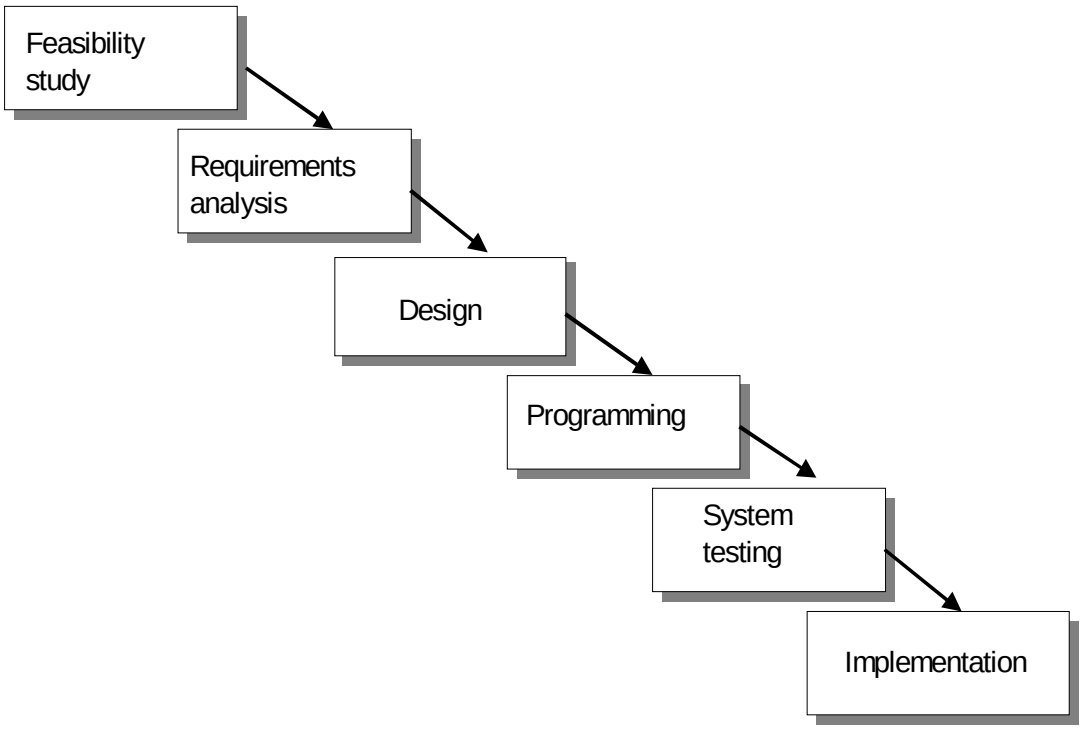

Figure 2. Waterfall Life Cycle

A number of variations of the waterfall life cycle are in use today. Depending on the nature of project, one can choose from pure waterfall, spiral, prototype, staged, or some form of rapid application development (RAD). A description of these alternatives can be found in any software engineering textbook. Most textbooks also offer advice for choosing the appropriate development process for a specific project.

Defining the Work

The basis for all planning activities is the Work Breakdown Structure or WBS. It decomposes the project into hierarchically structured well-defined, manageable tasks or activities. A WBS can be in the form of a table or a chart. The example of the chart form in Figure 3 shows the hierarchical relationships among the various tasks of the project. The number of levels of detail depends mostly on project size and personal preference of the project manager. It is important that all activities necessary to complete the project be included in the WBS and assigned to an individual or a specific organization in an unambiguous manner. The WBS provides the fundamental framework for scheduling, budgeting and project control. Once the WBS has been defined, the estimating process can start.

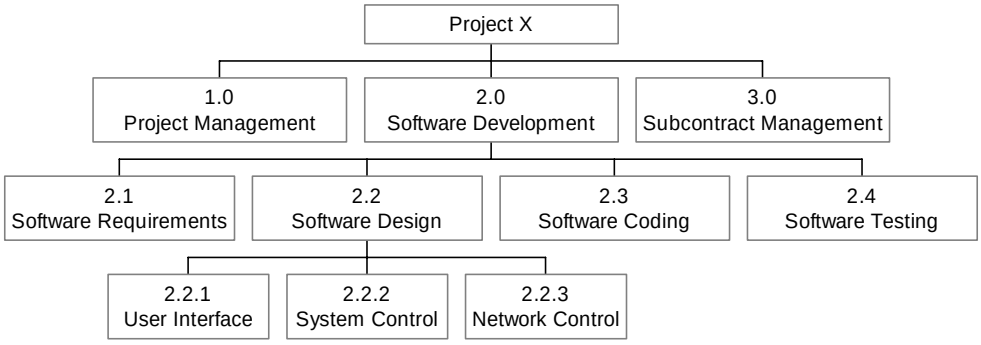

Figure 3. Work Breakdown Structure (WBS)

ESTIMATION

Software Size Estimation

The first step is to estimate the size of the software to be developed. The quality of this estimate directly influences the cost and schedule estimates. It is also the most difficult part of the estimation process. It is often done poorly, as evidenced by many cost overruns and schedule slippages.

Two measures of product size are commonly in use: lines of code and function points. The lines-of-code approach requires an estimate of the number of lines of source code, typically estimated by analogy with similar past projects. Function point analysis, on the other hand, does not require prior knowledge of source code. It is based on a synthetic measure of business functionality. The number of function points is determined from a weighted sum of inputs, outputs, inquiries, master files, and interfaces with other programs. The raw function point count is adjusted for technical complexity and environmental factors in arriving at a final function point estimate. Function points lead to reasonably reliable estimates and are independent of programming languages. Almost all automated commercial software estimation tools support function point analysis. Complete descriptions of function point analysis can be found in Dreger (1989) and Jones (1991).

Cost and Schedule Estimation

The two fundamental approaches to software project cost and schedule estimation are top-down and bottom-up. The top-down approach attempts to estimate the total project cost and schedule, typically using automated software cost estimation models. This process consists of converting the size measure to effort in terms of staff-months and to project duration in terms of days or months. Although various algorithms and rules of thumb are available, all estimates need to be adjusted to organizational productivity and other influencing factors. The over-all size estimate is then used to allocate the effort into specific tasks and activities and to schedule milestones.

The bottom-up approach starts with estimating the staffing needs and schedules for the lowest tasks and aggregating them to higher-level estimates and milestones. The top-down approach leads to superior estimates (Jones, 1996) while the bottom-up approach tends to instill ownership and commitment to the plan at all levels of the project team (Larson and LaFasto, 1980).

To obtain the best of both methods, many companies use the two approaches together in an iterative fashion. The top-down approach is used to define guidelines for the project as a whole, while a bottom-up approach is used to develop detailed cost and schedule estimates within the constraints established by the top-down approach. This method requires several estimating cycles before converging on a satisfactory estimate.

Commercial software estimating tools produce nominal schedules achievable by an efficient team working under average conditions. Many tools also provide capabilities for making schedule and cost trade-offs. The schedule can be shortened by adding more staff, but unlike other types of projects, software development does not allow for significant schedule compression. Research shows that practically all efforts to compress the schedule by more than 25 percent from the nominal are not successful (Boehm, 1981).

A good schedule is challenging, but it must be reasonable and achievable. It must also have the commitment of the whole project team. The best way to obtain commitment is to have each task estimated by the individuals or groups responsible for completing it.

The schedule for large projects is usually divided into a master schedule, showing only major activities and milestones, and supporting lower tier schedules for detailed activities.

The most common form of presenting schedule information is the Gantt chart that portrays activities against a horizontal time scale (Figure 4). Gantt charts are popular because they are well understood and easy to create and revise. However, Gantt charts are not well suited for showing the interrelationships among the various tasks (which task has to be finished before another one can start). A schedule network is the appropriate tool for showing interrelationships. The most commonly used schedule networks are PERT and CPM (critical path method). Both methods are quite similar in their use of flow charts to show the interrelationships among tasks. A sample schedule network is shown in Figure 5.

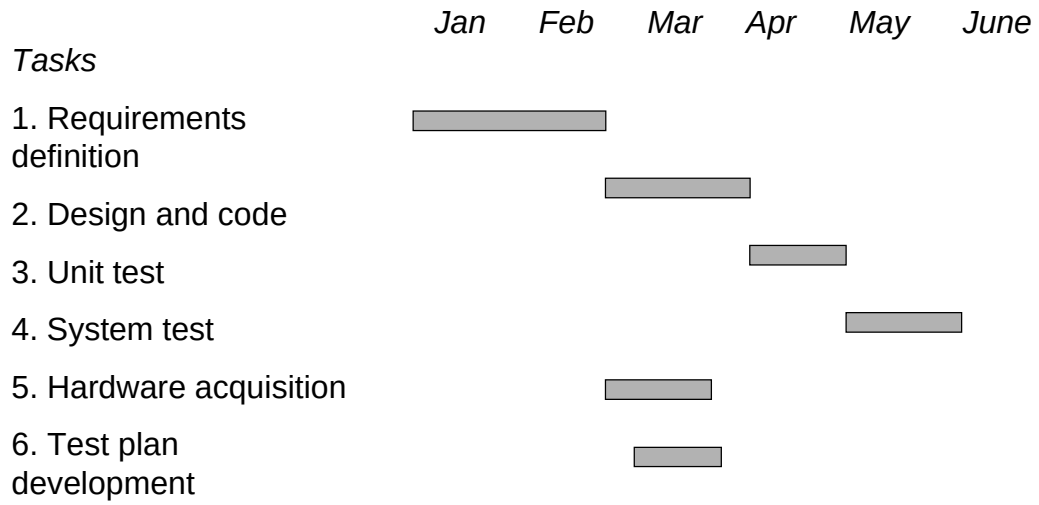

Figure 4. Sample Gantt Chart

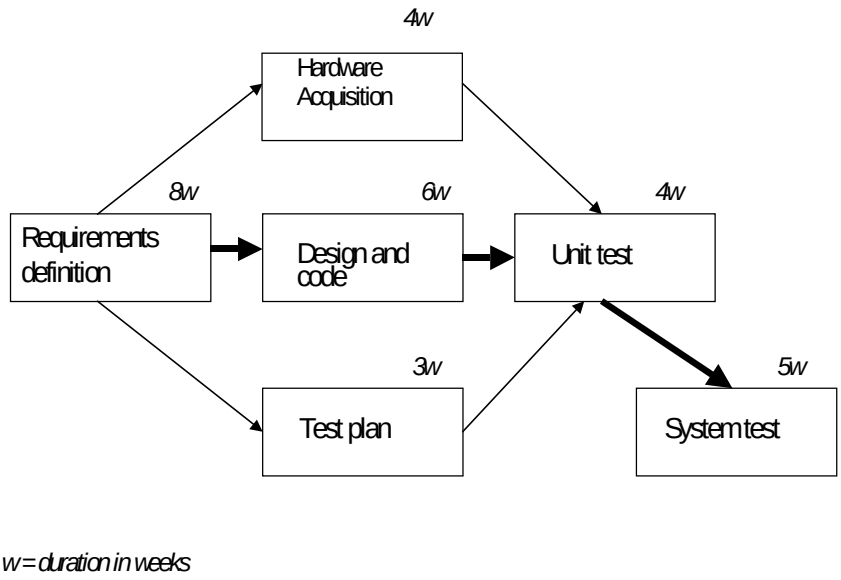

Figure 5. Sample Schedule Network

Another advantage of schedule networks is that they can be used to identify and track the critical path of a project. The critical path is the set of activities along the path that takes the longest time to complete (shown as the bold line in Figure 5). In essence, the critical path determines when the project will be completed. The tasks on the critical path require special management attention because any schedule slippage in these tasks leads to a corresponding slippage in the project completion date. It should be noted that a project can have multiple critical paths and that variations in the duration of tasks can shift a critical path from one set of activities to another. The critical path is also useful for identifying serious schedule risks (see below). Practically all project management software tools include the means of creating and displaying both Gantt and schedule networks.

Cost Estimation and Budgeting

The remaining estimation tasks are cost estimation and budgeting. Software project costs are driven mainly by staffing costs. The number of staff-hours can be derived directly from the project size estimates with the aid of

automated estimating tools or by using the company's own historical database. In either case, the estimates must be adjusted to match the capabilities of the team, its experience, and skill levels. The estimate should cover all activities that are identified in the work breakdown structure, including project management and all support functions, such as quality assurance.

In addition to direct labor costs, the estimate must include all direct non-labor costs, and overhead or indirect costs. Direct labor costs are determined by multiplying staff-hours by appropriate labor rates for each WBS item. Direct non-labor costs are all other charges applied to the project, including tasks that are outsourced, consultants, travel, and material costs.

The total project budget is determined by aggregating all direct and indirect costs. It is a good practice to include some management reserve for unanticipated problems and contingencies. A reserve of 5 to 10 percent is not unusual and it may be even higher for high-risk projects. The budget (with the contingency funds removed) is allocated to organizations or individuals assigned to the project according to the WBS. The budget becomes the baseline for project control by providing standards against which project performance can be measured. A useful tool for project managers is a time-phased cumulative cost curve, sometimes called the s-curve. It provides visibility by representing the budget graphically as a function of time according to the project schedules (Figure 6.)

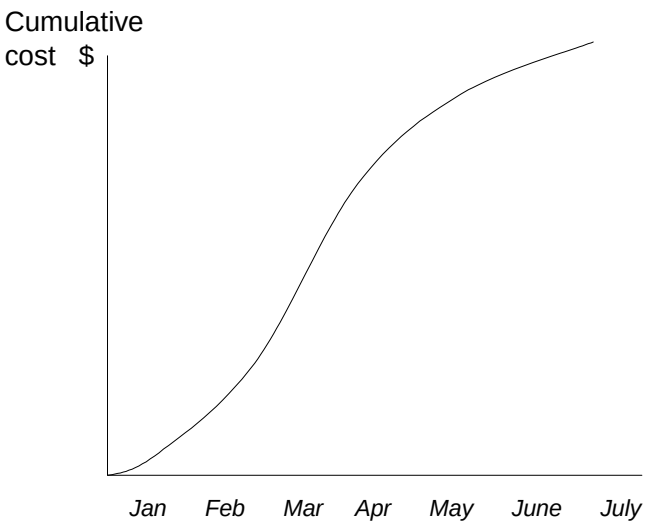

Figure 6. Cumulative Cost Curve

RISK MANAGEMENT

Risk management is an important part of the project management function. In the planning phase the project manager needs to perform a realistic assessment of risks and develop a plan for controlling these risks. Three major factors that influence project risk are:

- project size,
- project structure, and
- experience with technology.

The larger the project, the less structured it is (i.e. the requirements are not well defined and are likely to change during the project), and the less experienced the team with the technology of the project, the greater the risk. McFarlan (1981) recommends a contingency approach of adopting an appropriate project management strategy for each type of risk (Figure 7). A set of management tools is available for implementing each strategy. They include external integration tools, internal integration tools, and formal planning and control tools.

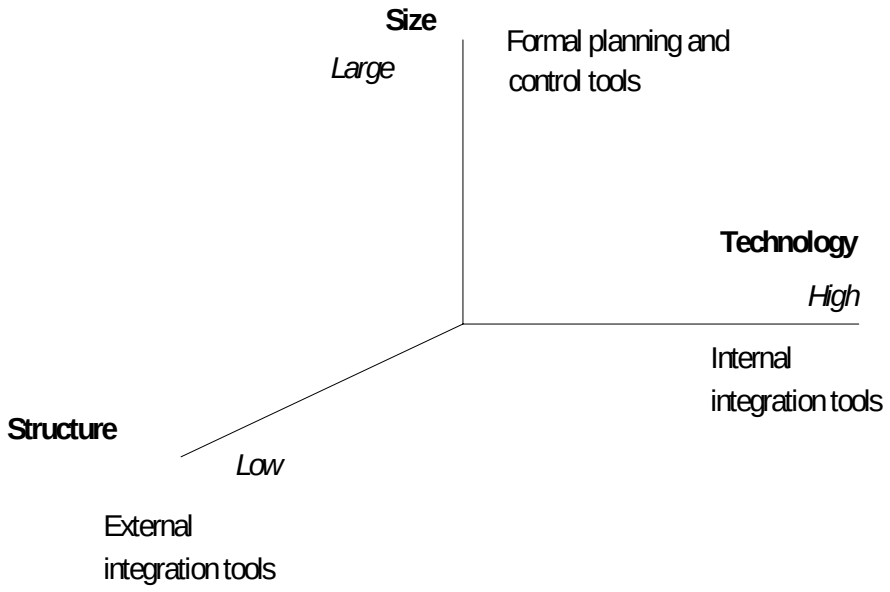

Figure 7. Contingency Approach to Risk Management

Projects with relatively little structure can benefit from external integration tools that create effective links between the project team and the client's organization. Examples of such tools include:

- Selecting the project manager and key team members from the client's organization
- Frequent client representation at project review meetings
- Wide distribution of status reports in the client organization

Projects involving new technology should rely more on internal integration tools that are designed to enhance the team's technical competence and operation as an integrated unit. Typical internal integration tools include:

- Highly experienced team members
- Project manager with a strong technical and project management background

- Frequent team status meetings

Large projects, particularly those with high structure, should be managed by formal planning and control tools. These tools, described earlier in Sections III and in Section IV, represent a highly disciplined, systematic approach to project planning and control. They include WBS, formal schedules, budgets, and tracking procedures for management control.

While the choice of appropriate management approach for dealing with project risk at high level is important, a more detailed risk assessment is needed to address specific potential risks. Risk assessment includes

- risk identification,
- risk analysis, and
- risk prioritization.

The purpose of risk identification is to develop a list of risks that can adversely impact project outcome. Risk analysis consists of assessing the risk exposure, the likelihood and impact of each risk. Risk prioritization produces a list of risks prioritized by impact that becomes the basis for risk management planning. This plan addresses all major risks, identifies contingency plans for dealing with them, and defines the process of monitoring the risks. A useful tool for risk monitoring is a top-10 risk list that is frequently updated to identify the ten highest risks in the order of their priority.

SOFTWARE PROJECT PLAN

Project planning culminates in a software project plan. It is a document that describes the overall approach to software development, specifies all deliverables, resource requirements, schedules, budgets and organizational responsibilities, and defines the management processes. In addition, it outlines all risk factors and risk management strategies, and describes how changes are managed and quality is assured. It is a document that informs management, team members, and the client. It is like a roadmap that serves to guide the project team members. It establishes the cost and schedule baseline for

managing and controlling the project. Therefore it becomes an effective part of the project control system. To test the adequacy of a project plan, the project manager should ask:

- Does the plan allow me to manage the project effectively?
- Does it provide enough information for the team members to plan and do their work?
- Does it have the commitment of senior management and the project team?

A typical software project plan is outlined in Table 2.

IV. PROJECT ORGANIZATION

SELECTING THE PROJECT MANAGER

The project manager is usually selected at the end of the conceptual phase when the project is approved. The project manager is the person who is responsible for managing the entire project. His or her primary responsibility is to direct and coordinate all activities to meet the objectives of the project within budget and schedule. This role is quite different from the role of the technical leader or developer, whose responsibility is mainly for the technical integrity of the product. Specific responsibilities of the project manager include the following:

- Reporting to senior management
- Communication with users
- Planning and scheduling
- Coordinating project activities
- Budget, schedule, risk, and quality control
- People management
- Delivering results

Table 2. Project Plan Outline

1. Introduction

1.1 Project Overview
1.2 Project Deliverables
1.3 Evolution of the Software Project Management Plan
1.4 Reference Materials
1.5 Definitions and Acronyms

2. Project Organization

2.1 Process Model
2.2 Organizational Structure
2.3 Organizational Boundaries and Interfaces
2.4 Project Responsibilities

3. Managerial Process

3.1 Management Objectives and Priorities
3.2 Assumptions, Dependencies, and Constraints
3.3 Risk Management
3.4 Monitoring and Controlling Mechanisms
3.5 Staffing Plan

4. Technical Process

4.1 Methods, Tools, and Techniques
4.2 Software Documentation
4.3 Project Support Functions

5. Work Packages, Schedule, and Budget

5.1 Work Packages
5.2 Dependencies
5.3 Resource Requirements
5.4 Budget and Resource Allocation
5.5 Schedule

6. Additional Components

7. Index

8. Appendices

* Based on IEEE Std 1058.1-1987.

The project manager's job is to provide visibility and manage commitments. Finding a person who can handle these challenges successfully is not easy. Few people have the qualifications and attitudes necessary to succeed in managing complex projects. It is even more difficult to find an experienced project manager who has the right qualifications and who is available for a new project. Good project managers are always busy on existing projects. Many companies create project or program offices to provide better management oversight and develop a pool of future project managers.

Effective project managers should have the following skills:

- Communication
- Organization
- Team-building
- Leadership
- Negotiation
- Goal orientation
- Ability to work under pressure
- Technical competence

Having a certain level of technical competence is helpful, but managerial and interpersonal skills are the most important for project managers. Broad background is more important than expertise in any technical area. Successful project managers are generalists, not technical specialists. They can come from various parts of the organization, not necessarily from the IS organization.

SELECTING TEAM MEMBERS

Software is like sports. The difference between the most productive and least productive programmers is huge. Therefore projects should select the best. The skills to look for are not only technical skills, but also problem solving and interpersonal skills. Good team members have high self-esteem and strong commitment to the project's success.

Project managers can turn to personality indicators based on Meyers-Briggs tests to create teams with a balanced mix of personality types. Many programmers are introverts and thinking persons who base their decision on facts rather than on feelings and personal values. They often find it difficult to build relationships and see the project from the user's point of view. Forming a balanced team with a variety of personality types can make a team more successful.

STRUCTURING THE TEAM

The way teams are organized has an enormous effect on how efficiently they perform. An inappropriate team structure can lead to longer development time, high cost, poor quality, poor communications and morale, and high turnover which, in turn, can lead to cancellation of the whole project. No one structure is appropriate for all projects. A structure that fits one project may be disastrous for another. The following four team structures have been used for software development projects.

Isomorphic Team

An isomorphic team is organized along the structure of the main deliverable software modules, as shown in Figure 8. Each team member is assigned to work on a specific software module from the beginning to the end. The advantages to this structure are:

- It is organizationally simple
- It allows many tasks to be developed in parallel
- Task responsibilities can be clearly defined and understood

Isomorphic teams are well suited for projects where different software modules are relatively independent of each other. A major disadvantage of this organizational structure is that it can lead to serious difficulties in integrating modules.

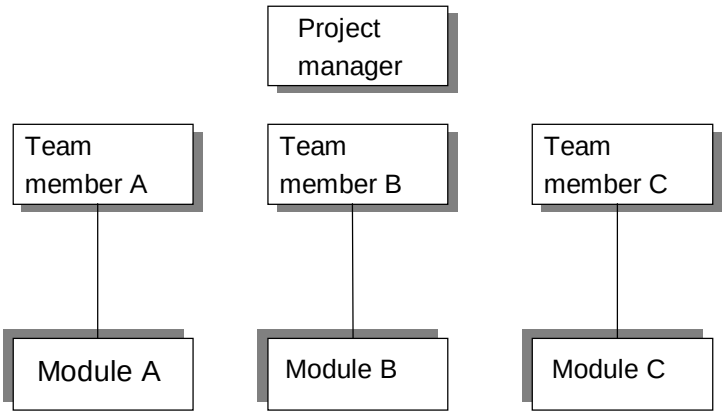

Figure 8. Isomorphic Team Structure

Specialty Team

In this structure each team member applies his or her special expertise across many software modules as shown in Figure 9. The primary advantage of this structure is that it allows special expertise to be used most effectively. Disadvantages include difficulties in establishing accountability and problems with integration.

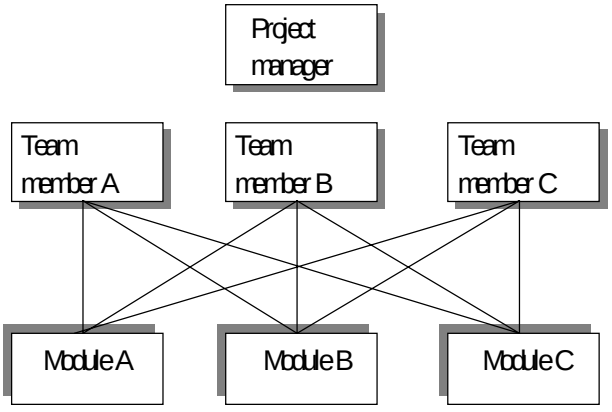

Figure 9. Specialty Team Structure

Egoless Team

This type of team structure, sometimes-called democratic programming team or self-managed team, does not have a formal structure (Figure 10). Decision making is shared among the team members. The structure encourages communication and interaction among team members. It works only in certain situations, mostly for small, ill-defined development projects where innovation and creativity are more important than meeting tight deadlines.

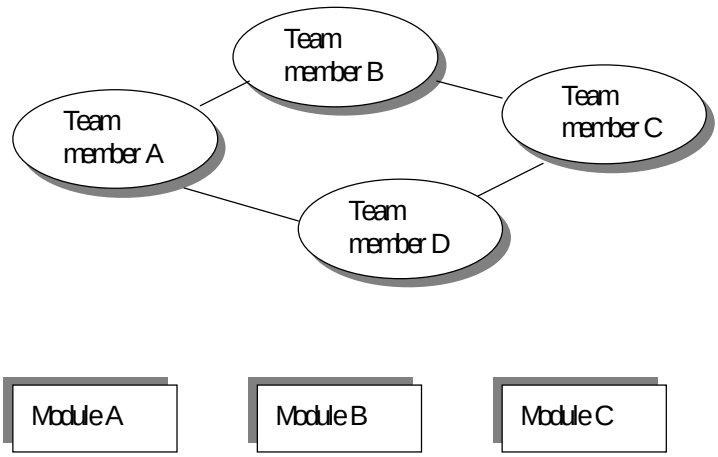

Figure 10. Egoless Team Structure

For most software projects egoless teams are not effective. They fail primarily because people, particularly highly talented software developers, do have egos. Another problem with egoless teams is that they tend to drift because they lack leadership.

Chief Programmer Team

This structure was developed at IBM for dealing with complex software development projects. The structure is diametrically opposite to egoless team structure. In this approach, all important decisions are made by the chief programmer who is supported by various specialists performing detailed support functions assigned by the chief programmer (Figure 11). This approach is conceptually similar to surgical team structure in hospitals where the surgeon performs surgery on the patient, supported by a team of specialized assistants.

In software development, the chief programmer is backed up by an assistant chief programmer, who works closely with the chief programmer and is able to take over in the chief programmer's absence. The project manager's role in this case is that of an administrator and a resource provider.

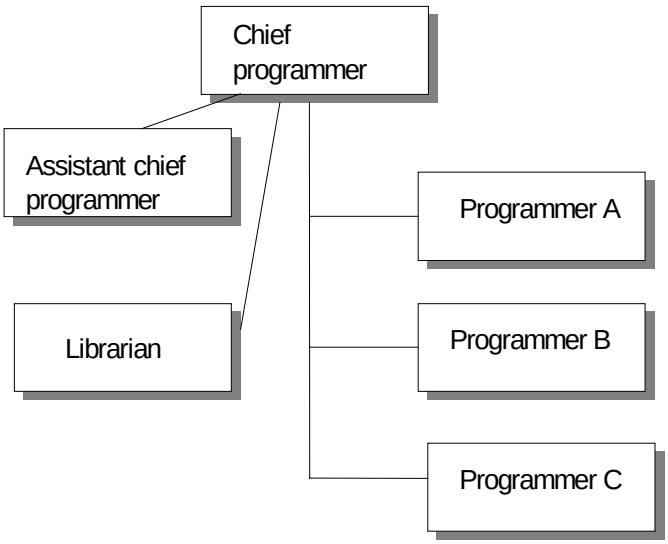

Figure 11. Chief Programmer Team Structure

V. PROJECT CONTROL

PROJECT CONTROL PROCESS

The purpose of project control is to

- keep the project on course and as close to the plan as possible,
- identify problems before they happen, and
- implement recovery plans before unrecoverable damage is done.

It involves comparing progress with the plan and taking corrective action when performance deviates significantly from the plan. It serves as a feedback function, shown in Figure 12.

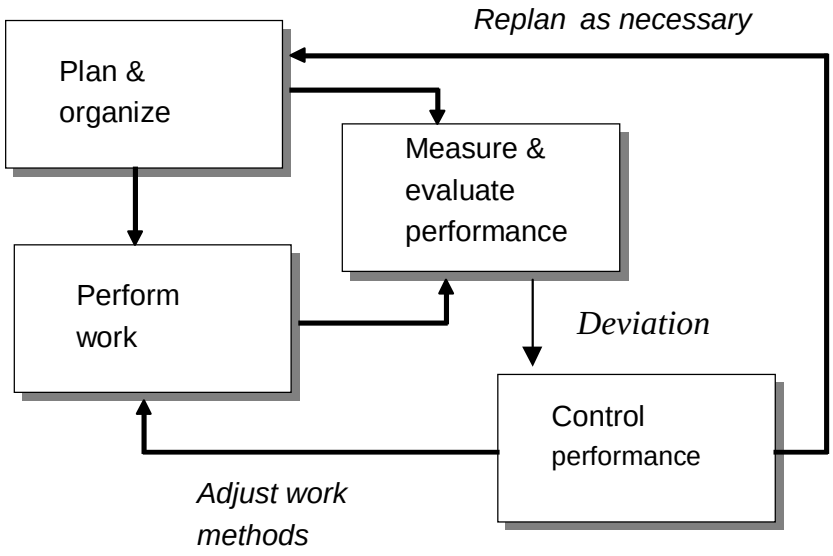

Figure 12. Project Control

For timely and effective control, the project manager must have full visibility of progress. Therefore, the project must be tracked and monitored systematically. The degree of formality and the frequency of monitoring depend on the type of project. The basic variables to be tracked are cost, schedule, and technical performance.

COST AND SCHEDULE CONTROL

Cost and schedule tracking involves comparing current status against the project baseline schedule and a time-phased budget that shows how budgeted costs are distributed across various activities. Any significant deviations or variances from the plan require prompt project manager attention so that timely corrective action can be taken. Knowing that the project deviates from the plan is not enough. The project manager must be able to identify the source of the problem. If there is a major deviation from the plan, the project manager must decide whether replanning future activities is warranted.

To obtain sufficient visibility for these decisions, cost and schedule status is analyzed in an integrated manner. Being within budget is not meaningful if the

tasks are not completed on time. Similarly, completing tasks as scheduled is not noteworthy if accomplished at a significant cost overrun. The WBS provides the fundamental framework for comparing cost and schedule status against the project plan.

Earned Value Technique

Even though Gantt charts and cost accumulation reports provide useful indicators of project status, they have some limitations. The primary limitation is that they are complex and difficult to analyze, particularly for large projects with many tasks. This complexity can lead to distorted perception of project status. A technique for providing a better, more holistic view of progress, is the earned value approach. Originally developed for better tracking of large-scale government projects, it is nowadays used for many commercial software projects.

Earned value is a hybrid measure that expresses the value of completed work in terms of the budget assigned to that work. It allows separating the variance from the plan into two components: cost and schedule variance, both expressed in monetary terms. The cost variance is due to the variation in the price of the work done while the schedule variance is due to work done at a different time than scheduled. Earned value and the variances can be calculated for a single activity, a group of activities, or for the whole project. From the cost and schedule variances it is possible to determine the project's cost and schedule performance indices. These indices provide instant visibility to the project’s performance. They allow the project manager to estimate the cost at completion and the actual completion date, based on performance to date. Most project management software systems (e.g. Microsoft Project) support earned value calculations at both total project and sub-project levels. While an extremely powerful and essential tool for large-scale projects, the technique can be equally useful for smaller software projects. The essentials of earned value calculations are described in the Appendix I.

Project Review Meetings and Status Reports

Review meetings play a major role in project control. Their purpose is to assess progress and identify areas of deviations from the plan so that corrective action can be taken. They are a mechanism for openly discussing current and potential future problems and communicating among team members. Project review meetings provide visibility to plans and progress and create opportunities for obtaining and enforcing commitments from the participants.

Project review meetings are most effective when they are scheduled at regular time intervals (weekly for most projects), and follow an established agenda. A typical agenda might include the following:

- Status of high-risk areas
- Overall project progress (major milestones, schedule, cost)
- Progress of specific activities
- Status of action items
- Future planning

They should be attended by appropriate representatives from each major area who can adequately answer questions, negotiate solutions, and make commitments.

Project reviews are different from technical and quality reviews, in that the latter are designed to detect and correct technical and quality problems rather than review progress. They are also different from management review meetings that are held with a senior management team or a steering committee at less frequent time intervals.

Status reports are prepared and sent regularly (via e-mail) to inform people outside the project organization. To be effective, they must be kept short and timely.

TECHNICAL PERFORMANCE CONTROL

Technical performance control, the process of assuring that all technical requirements are met, is normally exercised through a variety of design reviews. These reviews are usually held at major milestones (e.g. completion of requirements definition phase, design phase, or coding) but can be held at other times during the project. The purpose of design reviews is to show actual achievement or prediction of certain key technical objectives. The progress toward important technical goals should be tracked through appropriate metrics during the project. The metrics provide project managers visibility of what has been achieved, and their trends offer predictions of what can be expected in the future. A useful practice is to have the client's representatives participate in technical reviews to assure a common understanding of client needs and avoid future surprises.

QUALITY ASSURANCE

Quality assurance (QA) makes sure that the product meets user requirements and that it provides the desired functionality and quality. Quality, like technical performance requirements, should be defined in specific, quantifiable terms that are well understood by the client and the project team. While the whole project team should be committed to building quality into the product, it is a general practice to have a separate individual or a group whose primary responsibility is quality assurance.

The main purpose of QA is to detect and correct errors as early as possible. Early detection and correction of errors can result in significant cost and schedule benefits. The cost of correcting errors in the design phase is about one tenth of the cost of correcting them in the testing phase.

The basic tools for quality assurance are technical reviews and testing. Technical reviews are effective because they find defects early and tend to find different types of errors than testing. The most common types of reviews are walkthroughs and inspections. Walkthroughs are relatively informal reviews at

which several team members review the design or code to identify problems and improvements. Inspections are more formal reviews where reviewers use checklists to stimulate the review and use formal record-keeping and systematic feedback to improve the development process. Both types of reviews are effective for early detection of errors in requirements, interface prototypes, design, code, and documentation. Testing, the most common QA practice, is the systematic exercise of programs to find defects that were not detected earlier. Conducted at both unit and system level, they are used to verify the functionality and quality of the system.

To track the effectiveness of the QA process, a number of different metrics can be used. Two commonly used QA metrics are defect density (the number of defects per 1000 lines of source code) and defect-removal efficiency (the percentage of defects removed by any given operation, such as code inspection or testing). By applying these metrics to each specific software module and defect type, they become powerful tools for project managers in quality tracking and analysis.

CHANGE AND CONFIGURATION CONTROL

Change control is simple in concept, but complex in detail. Even the best prepared requirements specifications will require changes as the software is being designed and tested. Many projects fall victim to "scope creep" caused by uncontrolled changes made well beyond the requirements definition phase. A central control mechanism with automated configuration control tools and a strong change control board is essential.

Figure 13 shows a typical change control process. It starts with a change request, also known as an engineering change proposal (ECP), that identifies the need for change, the nature of the change, and the impact of the change. It is submitted to the change control board for review and disposition. The change control board, usually chaired by the project manager or his designate, consists of representatives from each area that has a stake in the system development. It

is designed to guarantee that all parties affected by the change will understand the consequences of each change before making it. After an appropriate review, the change control board must ensure that:

- only necessary changes are made,
- changes are made in a controlled fashion, and
- changes are communicated to all parties involved.

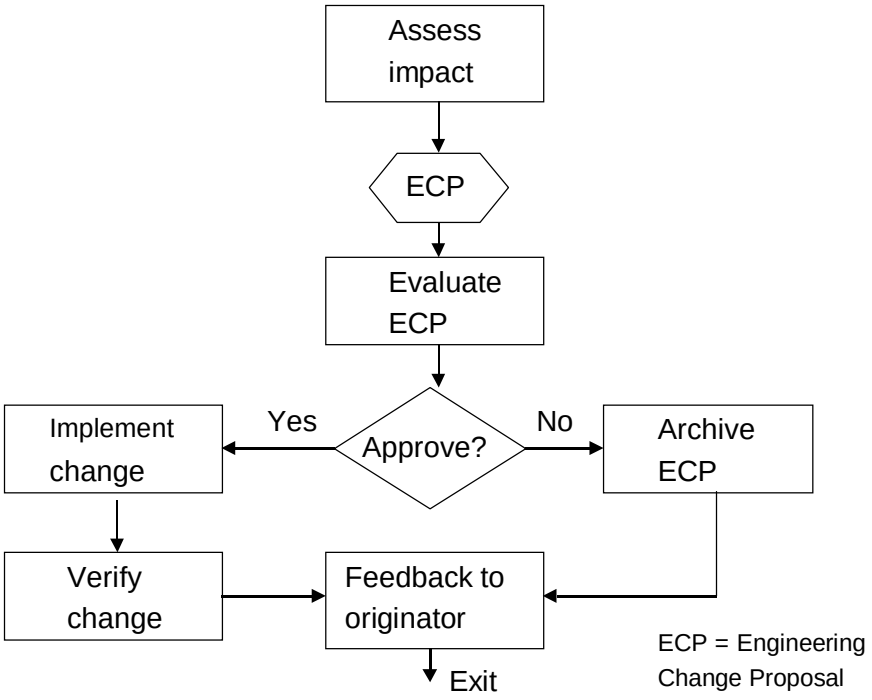

Figure 13. Change Control Process

PROJECT EVALUATION

Project evaluation is the process of periodic assessment of project status relative to its goals. It differs from project control in two aspects:

- Project control is the responsibility of the project manager, but project evaluation is a task for senior management.

- Project control is a continual process, while project evaluations take place only periodically.

Typically evaluations are performed at some important milestones to determine whether major changes are warranted. These changes may include reassessment of the goals and objectives, restructuring of the project plan, or even project cancellation.

Project evaluation also has a major role at the completion of the project. In this case the objective is to evaluate past experience and develop lessons learned for the benefit of future projects.

VI. PROJECT MANAGEMENT THROUGH TEAM BUILDING

MOTIVATION AND TEAM BUILDING

Team building and development is usually covered in management and behavioral sciences literature but is often ignored in software project management texts and courses. Many software project managers tend to focus their primary attention on technical matters while ignoring people issues. By doing so they put their projects at great risk. Major project failures often can be traced to dysfunctional team performance caused by inadequate attention to people and teamwork issues.

Team building is the process of transforming a collection of individuals from different backgrounds into a cohesive high performance team and keeping it motivated and focused toward the goal. It is an ongoing process that requires leadership skills and an understanding of many organizational factors. High performance teams tend to have many common characteristics. Chief among these are:

- A clear shared vision or goal.

A shared common vision is essential for a project to succeed. Research on successful project teams consistently demonstrates that a shared understanding of the project goal is the most important factor in project success. Agreeing on the project vision helps keep the team focused and productive. Decision-making is streamlined and time-wasting debates can be avoided.

- Commitment to the project.

Commitment can be defined as a sense of loyalty and dedication to the project. Committed team members are willing to devote their time and energy and make personal sacrifices for the project. Shared vision is the foundation for building commitment. Only when people understand and share the project's vision, are they willing to make a commitment to the project. Effective project managers work hard to build and foster commitment to the project. They look for ways to create exciting possibilities to make the project meaningful. They create stimulating work environments that provide interesting and challenging work with opportunities for professional growth. They build ownership by involving people in decision making. They make the project and the whole team's effort visible by keeping everyone informed of progress and how each team member's effort fits into the goals of the project. In effect, everyone becomes mutually accountable for the project, as a team.

- A strong sense of team identity.

Members of effective teams feel that they are special, that they are different from others. There is a sense of eliteness among the team members. Project managers can build and reinforce the sense of team

identify in many ways. A common practice is to use visible symbols to display team identity by putting the team's name or logo on t-shirts, caps, coffee mugs, and other items. Probably the best way to create a sense of belonging is to create a special work environment for the team by co-locating team members in single site or a dedicated work area. Meetings, beer and pizza parties, and milestone celebrations help reinforce a sense of belonging and team spirit. Successful project managers work consciously to create and shape unique cultures for their project teams to set them apart from the rest of the organization.

- Mutual Trust

Effective teamwork requires collaboration among team members. Collaboration can take place only in a climate of trust within the team. The greater the trust, the more likely the team will share information, report problems, and make effective decisions. Good project managers create and nurture an atmosphere of trust that builds confidence and commitment. Trust exists only in a team that values and rewards honesty and openness, where people are treated fairly, and with respect and dignity.

- Competent team members.

Effective project managers recruit competent people, based on the specific competencies needed on the project (Section IV). Collaboration skills for working effectively with others are just as important as technical skills. Project managers continually assess the performance of their team members and are willing to confront and reassign people with inadequate performance.

Project managers are also expected to manage relationships with the rest of the organization and remove obstacles. Team members are motivated when they know management is also committed to the project and is providing the necessary resources to the project. All this takes exceptional leadership skills that many project managers who have moved up through technical ranks may lack and need to develop.

Larson and LaFasto (1989) suggest the following leadership principles for team building:

- Avoid compromising the team's objectives with political issues
- Exhibit personal commitment to the team' goals
- Do not dilute the team's efforts with too many priorities
- Be fair and impartial toward all team members
- Be willing to confront and resolve issues associated with inadequate performance by team members
- Be open to new ideas and information from team members

COMMUNICATIONS

Many project failures can be traced to a breakdown of communications. It is the responsibility of the project manger to create an environment for effective communications within the team and manage the communication process with external stakeholders, particularly with the client's organization. Effective project managers keep all involved parties informed. They never surprise the client. They also do not depend on formal reporting structures alone. Body language at a status meeting can often provide more information than a carefully worded written status report.

CONFLICT RESOLUTION

Conflicts are part of project manager's daily life. They arise from problems within the team as well as from dealing with external stakeholders. Every project has abundant sources of potential conflict. The most common sources include competition for scarce resources, differences related to goals and the means to achieve them, and disagreements over cost, schedule or technical trade-offs. Some conflicts also arise from interpersonal relations. Regardless of the cause, project managers cannot ignore conflicts, they must identify them as they arise, understand the nature of their causes, and resolve them in their early stages. Failure to do so can seriously disrupt a project and lead to unnecessary delays and cost overruns.

VII. CRITICAL SUCCESS FACTORS

Much work has been done in using successful projects for benchmarking and identifying critical success factors (CSFs) for project management. CSFs are defined as those things that must go right for a project to succeed. Therefore they must be given special and continual attention from management to ensure success. CSFs have been applied in a variety of situations ranging from information systems planning to project management (Bullen and Rockart, 1986).

Project managers find CSFs particularly useful because most of their time is spent on dealing with a multitude of details and continuously “putting out fires.” As a result, they rarely have enough time to focus on issues that are less urgent, but critically important, to the success of the project.

The following CSFs, compiled from a variety of sources, can help project managers focus on areas that can make difference between success and failure in software development projects.

- Clearly defined objectives

  Complete and clear definition of project objectives, scope, and work to be performed.

- Top management support

  Senior management shows commitment by providing necessary resources, giving the project manager sufficient authority, and backing him or her in times of crisis.

- Adequate budget

  Many projects are doomed from the start because of inadequate budgets.

- Realistic schedule

  Too frequently, projects get off to a bad start because of overly optimistic schedules, often caused by unrealistic expectations.

- Client/user participation

  User involvement, particularly during the planning phase, leads to better and more realistic definition of system requirements and user commitment to the project.

- Project leadership

  Project leadership starts with the selection of the project manager and the key members of his or her management team. Effective leadership is needed to keep the team focused and motivated throughout the project.

- Project reviews

  Regular project reviews, held frequently, provide visibility into progress and problems. They also serve as a tool for sharing vision, motivating team members, and facilitating communications.

- Change control/management

Change control/management, the process of controlling and monitoring changes, is a challenge for all complex projects, but is particularly severe in information systems projects.

- Communications

Good communications among project team members and all affected parties is required. Many project failures can be traced to a breakdown of communications.

- Problem solving

No matter how well the project is managed, problems do occur. The success depends on an effective mechanism for anticipating and solving problems.

The relative importance of these critical success factors varies across different types of projects. Therefore, the primary value of the generic CSFs is that they provide a point of departure for project managers to develop their own set of factors, appropriate for their specific project needs.

The major advantages of CSFs are that they help project managers to:

- think through what is important,
- maintain focus on critical factors,
- establish priorities, and
- enhance communication and shared understanding.

The primary criticisms of CSFs are that they are not action oriented and they do not provide adequate guidelines for management action. However, CSFs can be used to develop appropriate performance measures that can become powerful

tools for managing projects. The process of arriving at these measures includes the following steps:

- Identify project goals and objectives
- Define and prioritize CSFs
- Develop a set of appropriate measures
- Implement a system for continuous monitoring of these measures

VIII. CONCLUDING REMARKS

Increasingly IS project managers find themselves playing a central role in their organizations, whether it is an enterprise resource planning (ERP) system implementation, Year 2000 conversion, or a leading-edge technology project. Many companies adopt a project management approach to many of activities outside the IS area. As we approach the new millennium, project management skills become more important than ever for career advancement. Experienced project mangers are in demand in variety of industries ranging from high-technology companies to financial services firms.

Business schools recognize this new emphasis on project management. They are introducing more formal project management courses and requiring students to work on projects that offer opportunities to gain the skills necessary for managing large and complex projects. Managing a successful project can be an extremely rewarding experience. In many ways it is like managing your own company. It is hard to imagine a better training ground for future senior executives than project management.

Editor's Note: This paper is based on a tutorial presented at the American AIS meeting in Baltimore in 1998. It was received on August 11, 199 and was with the author 2 weeks for one revision. It was published on September 30, 1999.

REFERENCES

Abel-Hamid, T. and Madnick, S.E. (1991) *Software Project Dynamics*, Enlglewood Cliffs, NJ: Prentice Hall.

Boehm, B. (1981) *Software Engineering Economics*, Englewood Cliffs, NJ: Prentice Hall.

Bullen, V.C. and Rockart, J.F. (1986) "A Primer on Critical Success Factors," in J.F. Rockart and C.V, Bullen (eds.) *The Rise of Managerial Computing*, Homewood, Illinois: Dow Jones-Irwin, pp. 383-423.

Dreger, B.J. (1989*) Function Point Analysis,* Englewood Cliffs, NJ: Prentice Hall.

Jones, C. (1996) “Software Estimating Rules of Thumb*,” Computer*, March, pp. 116 -118.

Jones, C, (1991) *Applied Software Measurement: Assuring Productivity and Quality,* New York: McGraw-Hill.

Kerzner, H. (1989) *Project Management*, Third Edition, New York: Van Nostrand Reinhold.

Larson, C.E. and Fasto, F.M.J. (1989) *Teamwork*, Newbury Park, CA: Sage Publications.

McFarlan, F.W., (1981) “Portfolio Approach to Information Systems,” *Harvard Business Review*, Vol. 59, No. 5,pp. 142-150.

BIBLIOGRAPHY

SOFTWARE PROJECT MANAGEMENT

Bennatan, E.M. (1995) *On Time, Within Budget: Software Project Management Practices and Techniques*, 2nd Edition, New York: John Wiley & Sons.

Brooks, F. P. Jr. (1995) *The Mythical Man Month*, Anniversary Edition. Reading, MA: Addison Wesley.

Carnegie Mellon University/Software Engineering Institute (1995) *Capability Maturity Model: Guidelines for Improving the Software Process* Reading, MA: Addison-Wesley,

Cusumano, M. A. (1997) “How Microsoft Makes Large Teams Work Like Small Teams,” *Sloan Management Review,* Fall, pp. 9-20.

DeMarco, T. (1982) *Controlling Software Projects*, New York: Yourdon Press.

Frame, J. D. (1995) *Managing Projects in Organizations*, Revised Edition, San Francisco: Jossey-Bass.

Gause, D.C. and Weinberg, G.M. (1989) *Exploring Requirements: Quality Before Design*, New York: Dorset House.

Gilb, T, (1988) *Principles of Software Engineering Management*, Wokingham, England: Addison-Wesley.

Glaser, G. (1984) “Managing Projects in the Computer Industry,” *Compute*r, December, pp. 45-53.

Kemerer, C. F. (1997) *Software Project Management: Readings and Cases*, Chicago: McGraw-Hill.

Mantel, S. Jr., J., & Meredith, J. R. (1995) *Project Management: A Managerial Approach*, Third Edition, New York: John Wiley & Sons, Inc.

McConnell, S. (1996) *Rapid Development,* Redmond, WA: Microsoft Press.

McLeold, G. and Smith, D. (1996) *Managing Information Technology Projects*, Cambridge, MA: Course Technology.

Phillips, D. (1998) *The Software Project Manager's Handbook: Principles that Work at Work*, Los Alamitos, CA: IEEE Computer Society.

Pressman, R. S. (1993) *A Manager’s Guide to Software Engineering*, New York: McGraw-Hill.

Putnam, L., and Myers, W. (1992) *Measures for Excellence*, Yourdon Press.

Rakos, J. J. (1990) *Software Project Management for Small to Medium Sized Projects*, Englewood Cliffs, NJ: Prentice Hall.

Royce, W. (1998) *Software Project Management*, Reading, MA: Addison-Wesley.

Whitten, N. (1995) *Managing Software Development Projects*, Second Edition, New York: John Wiley & Sons, Inc.

Yourdon, E. (1997) *Death March*, Upper Saddle River, NJ: Prentice Hall Inc.

Yourdon, E. (1992) *The Decline and Fall of the American Programmer*, Yourdon Press.

PROJECT PLANNING AND ESTIMATING

IEEE Standard for Software Project Management Plans (1987) The Institute of Electrical and Electronic Engineers, Inc., IEEE Std. 1058-1, New York.

Jones, C. (1995) "Determining Software Schedules," *Computer*, February, pp. 73-75.

Jones, T.C. (1998) *Estimating Software Costs*, New York: McGraw Hill.

Lederer, A. L. and Prasad, J. (1993) "Systems Development and Cost Estimating Challenges and Guidelines," *Information Systems Management*, Fall, pp. 3-41.

Putnam, L. H. and Ware, M. (1997) "How Solved is the Cost Estimation Problem?" *IEEE Software*, November/December, pp. 105-107.

SOFTWARE PROJECT RISK

Boehm, B.W., (1989) *Software Risk Management,* Washington*, D.C.:* IEEE Computer Society Press.

Charette, R.N., (1989) *Software Engineering Risk Analysis and Management*, New York: Intertext Publications.

Gogan, J.L., Fedorowicz, J., and Rao, A. (1999) "Assessing Risks in Two Projects: A strategic Opportunity and Necessary Evil," *Communications of AIS*, Volume 1, May, Paper 15.

SOFTWARE QUALITY CONTROL

Deutsch, M.S. and Willis, R.R. (1988) *Software Quality Engineering*, Englewood Cliffs, NJ: Prentice Hall.

Glass, R. L. (1992) *Building Quality Software*, Englewood Cliffs, NJ: Prentice Hall.

Kaplan, C., Clark, R., and Tang,V. (1995) *Secrets of Software Quality*, New York: McGraw-Hill.

Sanders, J. and Curran, E. (1994) *Software Quality*, Wokingham, England: Addison Wesley.

Weinberg, G. M., (1992-1996) *Quality Software Management*, Vols. 1-4, New York: Dorset House.

MOTIVATION AND TEAM BUILDING

DeMarco, T. and Lister, T. (1987) *Peopleware: Productive Projects and Teams*, New York: Dorset House Publishing Co.

Humphrey, W. S. (1997) *Managing Technical People*, Reading, MA: Addison Wesley Longman.

Katzenbach, J.R. and Smith, D.K. (1993) *The Wisdom of Teams*, New York: Harper Business.

APPENDIX I – THE EARNED-VALUE APPROACH

The earned-value approach is based on three basic parameters:

- *Planned budget*: Budgeted cost of work scheduled (BCWS)
- *Actual cost***:** Actual cost for work performed (ACWP)
- *Earned value*: Budgeted cost for work performed (BCWP)

Cost variance (CV) is calculated as:

CV = BCWP - ACWP

A negative variance indicates a cost overrun.

Schedule variance (CV) is calculated as:

$$SV = BCWP - BCWS$$

A negative variance indicates that the project is behind schedule.

Both variances are expressed in dollar terms. They can also be expressed in percentages as:

$$\% CV = CV / BCWP$$

$$\% SV = SV / BCWP$$

The *cost performance index (CPI)* is computed as:

$$CPI = BCWP / ACWP$$

Several methods based on CPI can be used to estimate the final project cost or *estimate at completion (EAC)*. A simplified formula for EAC is:

$$EAC = BAC / CPI$$

where BAC is the basic or budgeted cost at completion.

Similarly, it is possible to calculate the *schedule performance index (SPI)* as:

$$SPI = BCWP / BCWS$$

A simplified formula for estimating the project duration or *estimated time to completion (ETC)* is:

$$ETC = OD / SPI$$

where OD is the original duration.

APPENDIX II - PROJECT MANAGEMENT SOFTWARE

A large number of project management software packages are available on the commercial market. They help project managers plan and manage projects more effectively by tracking detailed activities and schedules and by providing top level visibility to progress. They do not replace experienced project managers. They only relieve project managers of the detailed clerical tasks and allow them to focus on more important aspects of the job.

Project management packages range from simple schedulers to enterprise-wide solutions and vary in price from about $50 to several thousand dollars. The fundamental features in most project management software packages include the following:

- Front-end planning and modeling

  Features for creating the initial plan and refining it by evaluating alternative plans. Most programs include PERT networks, critical path analysis, and provide resource leveling to even out the peaks and valleys of resource usage.

- Cost and schedule tracking

  Collecting cost and status data and tracking status against the plan.

- Reporting

  Preparing progress reports in various forms, including exception reports, Gantt charts, resource usage, earned value, and trend reports. Most packages let the users to create templates and customize the reports.

- Communications

  Assigning tasks and receiving status updates via e-mail and publishing reports on the company intranet site. Some companies also use Lotus

Notes® for communication and sharing project information with the project team.

- Multi-project management

Determine cross-project dependencies and generate multi-project reports to let managers assess the impact of individual projects across the business.

The following tools are typically provided as special software packages or are included in computer aide software engineering (CASE) toolkits:

- Configuration control

The ability to control and track changes, implement version control

- Software estimating

Models for software size, cost, and schedule estimating

Vendors are constantly introducing new features and adding functionality to their project management packages. Practically all new versions are Web-enabled, allowing project manager and team members to send data and review reports on company intranets and to share information with clients on extranets.

Vendors of popular project management packages include:

- Microsoft (Project 98)
- Primavera (Project Planner, Sure Track)
- Artemis (Project View)
- Scitor (Project Scheduler)
- Computer Associates (Super Project)

In selecting a project management software package, the following issues should be considered:

- How easily can the plans prepared and refined?
- How easily can the plans be updated?
- How well does it satisfy the reporting needs of the project manager?
- How well does it fit into the organization's business operations?

ABOUT THE AUTHOR

Jaak Jurison is Associate Professor and Area Chair for Information and Communications Systems at the Graduate School of Business, Fordham University, New York. He received his MBA degree and doctorate from The Drucker Graduate Management School of Claremont Graduate University in Claremont, California. He also holds BS and MS degrees in electrical engineering from Worcester Polytechnic Institute and Columbia University, respectively. Prior to entering academia, he was with Rockwell International Corporation, where he served in various management positions. Most recently he was program manager for a large multinational system development program for the Commonwealth of Australia.

Dr. Jurison's research interests include integration of business and technology, evaluation of information technology benefits, and international information technology issues. He has published numerous papers and co-edited two books: *Productivity in the Office and the Factory* and *Information Systems in a Global Business Environment.* He is currently on the Board of Directors of the Society for Information Management's Greater New York Chapter. He serves on the editorial review boards of *Communications of AIS, Journal of Global Information Management* and *Journal of Global Information Technology Management.*

LETTER TO THE EDITOR

DO RISK FACTORS AND SUCCESS FACTORS FOR SOFTWARE PROJECTS APPLY FOR ANY WORK SYSTEM, WHETHER OR NOT SOFTWARE IS INVOLVED?

Steven Alter
School of Business Administration
University of San Francisco
San Francisco, CA 94117, USA
alter@usfca.edu

Jaak Jurison's tutorial on software project management [Jurison, 1999] identifies three project risk factors (project size, project structure, and experience with technology) and also provides a list of critical success factors for projects. Why would one believe these are the most important risk factors for projects? For example, a *CAIS* article about project risk [Gogan, Fedorowicz, and Rao, 1999] published just five months before the tutorial looked at two cases and concluded that the three risk factors should be augmented by two others, time constraints and system interdependence. Furthermore, why would one believe that the tutorial's critical success factors for projects are the most important ones or even that they are uniquely about software projects? Is it possible that these success factors apply to almost anything done in organizations, regardless of whether software is involved?

This response is based on another previous *CAIS* article called "A General, Yet Useful Theory of Information Systems" [Alter, 1999]. That article argued that many phenomena related to information systems could be understood readily by viewing an information system as a special type of work system. A work system is a system in which human participants and/or machines perform a business process using information, technology, and other resources to produce products and/or services for internal or external customers. An information system is a work system that can only process information. A

project is a time-limited work system designed to go out of existence after producing a particular product.

Since information systems and projects are special types of work systems, any generalizations about work systems should also apply to information systems and projects. Similarly, generalizations about information systems and about projects should apply to particular types of information systems and to particular types of projects, such as software projects. Assigning generalizations such as risk factors and success factors at the most general level makes sense for the same reason that it makes sense to use inheritance in object-oriented programming, namely, clarity and the avoidance of unnecessary repetition. Applying the concept of inheritance to risk factors and success factors raises questions about which of Jurison's risk factors and success factors are about work systems in general, which are about projects in general, and which are specifically about software projects.

First consider the risk factors. The tutorial cites three of them that were proposed by [McFarlan, 1981].

- "Three major factors that influence project risk are: project size, project structure, and experience with technology. The larger the project, the less structured it is (i.e. the requirements are not well defined and are likely to change during the project), and the less experienced the team with the technology of the project, the greater the risk. McFarlan (1981) recommends a contingency approach of adopting an appropriate project management strategy for each type of risk."

In the context of a tutorial on software project management, these three risk factors refer to software projects. But was McFarlan actually referring to software projects or information system projects? Assume that the three major risk factors become part of the accepted wisdom of our field. Does this accepted

wisdom apply equally to software projects, information system projects, and work system projects? Doesn't it seem possible that information system projects and work system projects might have other major risk factors, such as management turmoil in the organization, lack of commitment, and lack of experience in organizational change? And when a relatively naïve individual (such as an MBA student without much experience) tries to learn about the field, wouldn't this accepted wisdom be misleading if it wasn't clear that the accepted wisdom was about software projects, but not necessarily information system projects or work system projects?

Related issues apply to the tutorial's list of ten critical success factors for software projects. At first blush it might seem surprising that none of the critical success factors are the opposite of the three risk factors. In other words, if project size, project structure, and experience with technology are important risk factors, then some version of these characteristics probably should also be mentioned in the list of success factors. (Or is there a difference between critical success factors and just plain vanilla success factors?)

Let's return to the idea of inheritance to see whether the ten CSFs in the tutorial are best described as CSFs for software projects, CSFs for information system projects, or CSFs for work system projects, Table 1 below lists each of these CSFs along with several corresponding success factors cited earlier in 1999 in five separate tables of success factors in Alter [1999].

- Table 4: Success factors for work systems
- Table 5: Success factors for information systems
- Table 6: Success factors for specific types of information systems
- Table 7: Success factors for projects
- Table 8: Success factors for specific types of projects

The success factors in each of those tables were placed at the most general level possible. In other words, a success factor that seemed to apply for work systems in general (and therefore for information systems, a specific type of

work system) were included in the Table for work systems. Also note that each success factor in each table was linked to a specific element of a work system, resulting in a large number of success factors.

Table 1 shows that the tutorial's ten CSFs for software projects correspond to some factors [Alter, 1999] had listed as success factors for work systems in general, others that were listed as success factors for projects in general, and yet others that were listed as success factors for IS projects.

A comparison in Table 1 between Jurison's success factors for software projects and Alter's success factors for work systems in general, projects in general, and IS projects shows:

- 3 of Jurison's 10 CSF's were listed as success factors for work systems in general (adequate budget, leadership, and communications).

- 8 of Jurison's 10 CSF's were listed (in only slight different form) as success factors for projects in general (adequate budget, project leadership, communications, problems solving, top management support, realistic schedule, change control/ management, client/user participation
- 4 of the 10 CSF's were listed (using more IS- and project-related terms) as success factors for IS projects in particular: (project leadership, communications, change control/ management, client/user participation, clearly defined objectives).
- 1 of the 10 CSF's (project reviews) appeared in the tutorial but did not have a corresponding success factor in the tables in Alter [1999].

Table 1: Comparison of success factors in [Alter, 1999] and [Jurison, 1999]

Phrase used in Jurison [1999]	**Level according to Alter [1999]**	**Phrase used in Alter [1999], Tables 4, 7, & 8**
Adequate budget	Work systems in general	• Adequate resources for business process [4] • Adequate technical infrastructure for the work system [4] • Adequate human infrastructure for the work system [4]
	Projects in general	• Management willingness to allocate necessary resources [7]
Project leadership	Work systems in general	• Effective operational management [4]
	Projects in general	• Consensus on project governance [7]
	IS projects	• Comfortable relationship between IT staff and work system participants and their management [8]
Communications	Work systems in general	• Ability to work together to resolve conflicts [4]
	Projects in general	• Culture of cooperation on projects [7] • Shared understanding of the project's goals, rationale, schedule, and resources [7]
	IS projects	• Comfortable relationship between IT staff and work system participants and their management [8]
Problem solving	Work systems in general	• Cooperative decisions about work methods [4]
	Projects in general	• Availability of subject matter experts (SMEs) who provide necessary knowledge about the situation [7]
Top management support	Projects in general	• Management commitment [7] • Management willingness to allocate necessary resources [7]
Realistic schedule	Projects in general	• Realistic expectations [7] • Confidence by project participants that the project can be done with the human and technical resources that are available [7]
Change control/ management	Projects in general	• Appropriate project management [7] • Attention to implementation in the organization [7]
	IS projects	• Adequately clear requirements for content and for plumbing [8] • Adequate external specification [8] • Adequate internal specification [8]
Client/user participation	Projects in general	• Customer involvement in designing and accepting the project's product [7] • Expert knowledge about the context and content of the work system being improved or created [7] • Informed agreement on the requirements [7]
	IS projects	• Inclusion of the overall information system effort the organization's plan [8]
Clearly defined objectives	IS projects	• Frozen requirements during programming and testing (unless the project involves use of a prototype) [8]
Project review	----------	Not included in Table 4, 7, or 8 in [Alter, 1999]

* The numbers in brackets refer to a specific table in [Alter, 1999]. Table 4 is success factors for work systems in general, Table 7 is success factors for projects in general, and Table 8 is success factors for information system projects.

The success factors listed in the tutorial certainly make sense and apply to software projects quite broadly. The purpose of this response was to note the possible advantages of separating out success factors that apply for work systems in general vs. projects in general vs. software projects in particular.

REFERENCES

Alter, S. (1999) "A General, yet Useful Theory of Information Systems," *Communications of AIS*, Vol. 1, Article 13, March 1999. http://cais.isworld.org/articles/1-13/

Gogan, J.L., J. Fedorowicz, and A. Rao. "Assessing Risks in Two Projects: A Strategic Opportunity and a Necessary Evil," *Communications of AIS*, Vol. 1, Article 15, May 1999. http://cais.isworld.org/articles/1-15/

Jurison, J. (1999) "Software project Management: A Manager's View, " *Communications of AIS*, Vol. 2, Article 17, September 1999. http://cais.isworld.org./articles/2-17/

McFarlan, F.W., (1981) "Portfolio Approach to Information Systems," *Harvard Business Review*, Vol. 59, No. 5,pp. 142-150.