SPM UNIT ONE *(as mentioned in the syllabus of TU BIM Eighth Semester)*

1. SOFTWARE PROJECT BASICS

INTRODUCTION

Human endeavor, from its earliest hunter/gatherer roots, was carried out in teams, each with a hierarchy of roles. As civilization progressed, the need for structure and rules increased. A large farm is a team organization based on a simple hierarchy of an owner, overseers, and employed laborers. The Industrial Revolution created factories which required more complex hierarchies, both within teams and between teams. Factories aggregated the production of goods for consumption into concentrated units capable of greater productivity. To achieve this great jump in productivity, rules were developed to effectively run the factories. These developments were the genesis of the art and science of managing production, which has been called *production management*.

Classification of organizations. The type of production can be used to classify organizations based on the *manner* in which goods are produced. The categories are:

- Mass production: continuously produces the same products
- Batch production: produces goods in batches; each batch is similar, but not identical
- Flow process production: production of chemicals, pharmaceuticals, and fertilizer products, generation of electricity, etc.
- Job order production: produces tailor-made goods (i.e., goods are produced only when an order is received)

Initially, management texts focused on *mass production*, *batch production*, and *flow process production systems* (also known as "made to warehouse" production systems). In *made to warehouse* production systems, goods are produced and stored in warehouses for distribution. The significant feature of mass production and flow process production is that the rate of consumption/demand *equals or exceeds* the rate of production for the product. In batch production, the rate of production *exceeds* the rate of consumption/demand for the product. The goal of production management is to balance both rates.

Production management texts, however, did not address organizations such as ship building, aircraft manufacturing, heavy equipment manufacturing, etc. These organizations are known as job order production or *made to order* organizations. In *made to order* organizations, items are produced only after an order is received.

By leaving out job order "shops," management texts also excluded organizations that constructed buildings, highways, and other infrastructure facilities. These types of organizations are certainly not serial production organizations even though they create wealth and employ people. Their work was classified as *projects*. Some knowledge, however, was gathered and released under the title of *project management*. Job order production system organizations latched onto this concept and became project-based production systems.

Presently, management theory addresses organizations in two basic categories: production organizations and service organizations. The art and science of managing these organizations has metamorphosed from *production management* to *operations management*.

Similarly, we can categorize organizations by the *nature* of their operations:

- Continuous operations: organizations with fixed facilities that carry out *similar* operations day after day continuously and produce products for stockpiling in warehouses (real or virtual)
- Project operations: organizations with fixed but flexible facilities that carry out *dissimilar* operations from day to day and produce only against a customer order

More and more organizations are moving toward project operations due to market forces, which put emphasis on individual preferences while reducing costs. Gone are the days of the famous words of Henry Ford, Sr.: "You can have the car of any color as long as it is black."

The project operations category has seen significant development over the past few years under the title "mass customization." Mass customization blends aspects of continuous and project operations.
Having put the concept of project operations in an historical perspective, see Table 1.1 for a comparison of continuous operations with project operations. Mass customization walks the line between the two extremes identified in Table 1.1, typically with most of the benefits of each, but with a greater reliance on self-directed teams that make hierarchies and matrix organizations very nervous.

Table 1.1. Comparisons of Continuous Operations with Project Operations

Item Number	Aspect	Continuous Operations	Project Operations
1	Product design	Designed once: updated as needed/dictated by market forces	Designed for every order received
2	Trigger for commencement	Marketing asks for the product	Customer's order triggers commencement
3	Planning	Periodic: annual, quarterly, monthly, weekly, etc.	Order-wise as well as periodic
4	Workstation design	Low cost: to produce one type of component	Potentially higher cost: versatile workstations to produce a wide variety of components
5	Required education levels for staff	Low: needs to understand instructions and can be easily trained (leads to a flatter training curve)	High: needs to be able to interpret drawings/ instructions and may require longer training (leads to a steeper learning curve)
6	Products	Batches of identical products	Products range from similar to (but never identical to) to radically different
7	Types of workstation operations	Mostly repetitive with little variety	Mostly nonrepetitive with wide variety
8	Specialization	Highly feasible	Limited specialization
9	Planning	Planning utilization of facilities becomes more predominant and important	Planning development of the product becomes predominant, while facility utilization planning becomes less important
10	Organizational structure	Hierarchical mostly	Mix of hierarchical and matrix organization
11	Customers	Repetitive customers possible to a high degree	Normally one-off customers with low probability of repeat order for same product

Description of a project. Let's now examine what comprises a project: a project is a temporary endeavor with the objective of manufacturing (producing or developing) a product or delivering a service, while adhering to the specifications of the customer (including functionality, quality, reliability, price, and schedule) and conforming to international/national/customer/internal standards for performance and reliability. Translation:

- A project is a temporary endeavor.
- A project has a definite beginning and a definite ending.
- No two projects will be identical, although they may be similar.
- Each project needs to be separately approved, planned, designed, engineered, constructed, tested, delivered, installed, and commissioned.
- A project may be stand-alone or a component in a larger program.
- A project is executed in phases, with an initiation phase and one or more intermediate phases and a closing phase.
- Many projects have a transition phase (e.g., handover to customer).
- A project may extend through a maintenance phase.

A software development project (often shortened to software project) has the objective of developing a software product or maintaining an existing software product. Software development projects have several general attributes, including:

- The project has a definite beginning and a definite end.
- The project deliverable is functional software and related artifacts.
- Activities that may be included in a project are user and software requirements, software design, software construction, software testing, acceptance testing, and software delivery, deployment, and handover.
- Activities not included in a project are the activities of project selection/acquisition and post-handover.

Some of the more unique attributes of software development projects include:

- The primary output is not physical — in the sense that the primary deliverable is functional software and no tangible components are delivered — almost everything is inside a computer.
- Process inspection does not facilitate progress assessment — functional software or at least the code is the real measure of progress. In a manufacturing organization, one can see semi-finished goods. The proof of work being performed is in the noise made by machines. In a software development organization, visual assessment is not enough to ensure that a person is performing. One needs to walk through the code being developed to ensure that the person is working.
- Despite significant progress in software engineering tools and diagramming techniques, they do not rise to the level of precision of the engineering drawings used in other engineering disciplines.
- Professional associations in software development and standards organizations have not defined standards or practices for developing software as has occurred in other engineering practices. The International Organization for Standardization (ISO) and the Institute of Electrical and Electronic Engineers (IEEE) have defined a number of standards, but these standards are not at the same level of granularity as other engineering standards.

- Although significant improvements in software development methodologies have been made, these methodologies are still largely dependent on human beings for productivity and quality. Tools are available to help in development or testing, but they still have not been able to rise to the level set by the standards and tools used in fabrication/ inspection/testing in other engineering disciplines. In other engineering disciplines, tools are available that shift the onus for productivity/ quality from human beings to the combination of tools and process. Most would agree that an average-skilled person can achieve higher productivity/quality with tools than a super-skilled person without tools.

Therefore, the rigor of planning is all the more important in software development than in other engineering projects — planning is a critical tool to keep a project focused. In other engineering projects, a simple schedule based on PERT/ CPM (Program Evaluation and Review Technique/Critical Path Method) would suffice, whereas in software development projects, increased rigor and more planning documents are required (planning documents commonly required are described in subsequent chapters).

TYPES OF SOFTWARE PROJECTS

Software development projects (SDPs) are not homogenous. They come in various sizes and types. Some examples will help us gain an understanding of the breadth of SDPs:

- An organization desires to shift a business process from manual information processing to computer-based information processing. This project will include studying the user requirements and carrying out all of the activities necessary to implement the computer-based system
- An organization desires to shift a business process from manual information processing to computer-based information processing. The organization does not want the software be developed from "scratch." It wants to use a commercial off-the-shelf software (COTS) product. This project will include implementation and perhaps some customization of the COTS product to make it appropriate for the organization.
- An organization has a computer-based system that needs to be shifted to another computer system because the existing system has become obsolete and support to keep the obsolete system in working condition is no longer available. This project could include porting the code, training users, and testing the new implementation.
- An organization has a computer-based system and desires to shift it from a flat file system to a RDBMS-based system (relational database management system). Activities will include data conversion in addition to other activities.
- An organization has a computer-based information processing system and needs to effect modifications in the software or add additional functionality. Activities include adding functionality and making required modifications in the software of a third party (if required).
- An organization has developed a computer-based information processing system and wants to get it thoroughly tested by an independent organization. Activities will include testing and interfacing between the organizations.

These examples barely scratch the surface of the breadth of software projects — and new project types keep coming in. In all cases, however, the projects concern software, but the tasks, activities, and therefore the work in each of the projects are vastly different.

CLASSIFICATIONS OF SOFTWARE PROJECTS

Software projects may be classified in multiple ways (Figure 1.1). For example, software projects may be classified as:

1. Software development life cycle (SDLC) projects
    - Full life cycle projects
    - Partial life cycle projects
2. Approach-driven software development projects
    - "Fresh" development (creating the entire software from "scratch")
    - COTS product customization/implementation
    - Porting
    - Migration
    - Conversion of existing software to meet changed conditions such as Y2K and Euro conversion

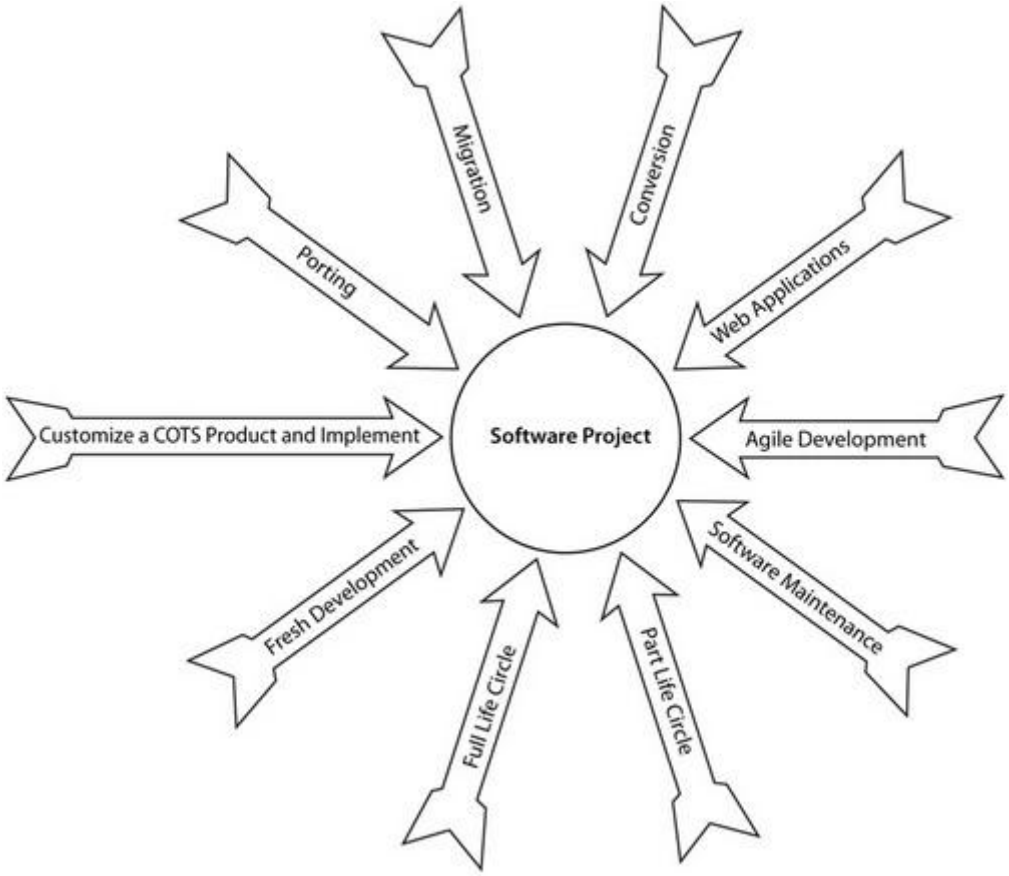

Figure 1.1. Software project types.

3. Maintenance projects
    - Defect repair
    - Functional expansion
    - Operational support

- Fixing odd behavior
- Software modification

4. Web application projects
5. Agile development projects

Let's now discuss each type of software project in greater detail.

1. Based on Software Development Life Cycle

a. Full life cycle projects:
A full life cycle project is a project that traverses the entire arc of the methodology being used: starts at the beginning and ends at the end. One problem when discussing a full life cycle project is that there is no standardization concerning what constitutes a software development life cycle (SDLC). Generally agreed is that user requirements analysis, software requirements analysis, software design, construction, and testing (regardless of what they are called) are parts of a SDLC. Some of the components of an SDLC that remain in question include:

- A feasibility study determining whether the project is worthwhile
- Special testing that is beyond unit testing, integration testing, system testing, and acceptance testing
- Implementation, including installation of hardware, system software, application software, etc.
- Software commissioning, including creating master data files, user training, pilot runs, parallel runs, etc.

In many instances, when the end product is used within the same organization, these four components are considered part of an SDLC. Alternately, in other circumstances, these components are excluded for organizations that specialize in software development and/or develop software for use by a different organization (unless contractually included or part of a software as a service architecture).

In this book, we exclude these four components. We assume that a full life cycle project is one that starts with user requirements and ends with the delivery of software. Therefore, all post-delivery activities and pre-user requirement activities are not considered to be within the scope of this book.

b. Partial life cycle projects:
Partial life cycle projects are those that include only a portion of the SDLC. In partial life cycle projects, any number of permutations could occur, including:

Testing projects in which the scope of the work involves conducting the specified or necessary software tests on the software product to certify the product (Unit testing and code walk-through are normally not included in this type of project.)

Independent verification and validation (IV&V) projects in which projects go beyond mere testing, including code walk-through and other forms of validation to determine the efficiency of coding

A project divided between two or more vendors based on the specialty to derive the advantages of best practices developed through specialization which can lead to defining the project by phase or by combination of phases, such as:

- Requirements analysis
- Design
- Software construction
- Testing

2. Approach-driven software development projects

   a. Fresh or new software development projects: Fresh or new software development projects are identical to full life cycle development projects previously discussed.
   b. COTS: product customization/implementation projects. Numerous popular COTS products are available in the marketplace. Examples include the implementation of ERP (enterprise resource planning software, e.g., by SAP and PeopleSoft), CRM (customer relationship management), SCM (supply chain management), EAI (enterprise applications integration), and data warehousing software. Typical phases in these projects include:
      - Current system study: a review of the present system
      - Gap analysis: a comparison of the current system to the COTS product
      - Customization report: a discussion of the desired levels of customization of the system
      - Statement of work: definition of the required customization of the COTS product
      - Design: how the software will accomplish the task
      - Construction and integration
      - Testing
      - Custom code integration: integration of the code bases (in some cases it can include building a layer over the COTS product and integration of custom developed code into the source code of the COTS product)
      - COTS source code modification (rare)
      - Implementation
      - Training: instruction of users (all classes required) in usage of the system, troubleshooting, and operations and maintenance of the system
      - Transition of the system

   Many variations of these phases are also possible for COTS projects.

c. Porting: Porting projects deal with moving software from one hardware platform to another hardware platform. Porting projects can include:
   - Changes in programming language
   - Differences between implementations
   - Manual intervention to make the existing software work on new hardware without issues

Project execution work in a porting project involves:

- Documenting the differences between the two versions of the programming languages
- Developing a software tool to make corrections in the code based on the details mentioned above (Sometimes, vendors of the programming language supply this type of tool.)
- Execution of the software porting tool to make all possible corrections
- Manual correction to make any specific corrections needed
- Conducting the specified software tests
- Modifications to the software engineering documents required to reflect the changes made in the software
- Conducting acceptance testing
- Delivery of the software

d. Migration: Oftentimes, new versions of programming languages and databases are released. For example, Visual Basic has gone through many versions: from version 1 to 6 and then the release of the next set as 2003, 2005, and 2008. Similarly, Oracle has gone through upgrades: up to version 11. Operating systems have also been upgraded. For example, Microsoft has had many upgrades including MS-DOS, Windows, 2 and 3, and then 95, 98, 2000, XP, Vista, and now Windows 7. When upgrades are released, upgrading software may become necessary:
   - To take advantage of new features and facilities provided in the newly released version
   - Because an older version is no longer available when additional hardware or system software is installed or the existing software does not function well on the new software (In these days of multitier Web-based software architectures, an upgrade of any tier may necessitate migration!)
   - Because limitations existing in an older version are removed in the new release and the existing software needs to be upgraded to remove the limitations
   - Upgrades are typically due to the ever-changing environment and the increasing needs of an organization. Of course, if the configuration of hardware and software remains exactly same, and the existing software is meeting the user's needs, the software would not need to be upgraded. A new version, however, could contain additional features and facilities that are totally absent in an older version. Therefore, a software tool cannot be used to make the changes that are necessary to port the software. To take advantage of new facilities and features available in the newer version, manual changes are typically required and involve:
     i. Studying the new version
     ii. Deciding which new features are desirable and need to be implemented

iii. Developing a functional expansion design document detailing the new features being implemented in the existing software
iv. Running and upgrading the software (if an upgrade tool is provided by the vendor)
v. Implementing the functional expansion design in the software coding and incorporating necessary software changes (may also include correcting the existing code)
vi. Conducting all the tests necessary to ensure that the software delivers all the functionality it was supposed to before migration and all the functionality that is designed for the new software
vii. Conducting acceptance testing and delivering the software
viii. Data migration involving (sometimes the project scope may include data migration):
    - Mapping the old database schema to the new database schema
    - Developing software/locating tools provided with the new database (if any) to migrate data from the old database to the new database
    - Running the tools to migrate data from the old database to the new database
    - Arranging for data entry in the new database for those fields that are absent in the old database, but present in the new database
    - Testing the database for known cases using the software, comparing the results with the desired results, and making necessary changes so that the new database is correct
    - Integrating the database with the software

Specific migration projects may have different activities from the activities described above. Note: Porting and migration projects are similar. There is no strict distinction between the two. Therefore, these two terms are sometimes used interchangeably.

e. Conversion. Year 2000 (Y2K) and Euro conversion projects are excellent examples of conversion projects. Using a Y2K project as example, the work includes verifying all programs for code limitations and then making any necessary modifications. Typical activities in a conversion project include:
    - Studying the existing software and specifications of the necessary conversion
    - Preparing a conversion guidelines document detailing the procedure for incorporating the required modifications in the software
    - Developing a tool (if feasible) to automatically incorporate the modifications in the software
    - Running the tools or hand coding the changes
    - Performing a manual walk-through of each program to locate the remaining required modifications and implementing them
    - Conducting unit testing (and other tests as specified or as necessary)
    - Conducting acceptance testing
    - Delivering the software

In Euro conversion projects, some countries that did not make use of decimals in their financial software had to incorporate decimals as well as provide for the use of the Euro symbol.

3. Maintenance

Software maintenance projects are major money makers for software development organizations that are dependent on outsourcing. We need, however, to relax the specific beginning and ending requirements to call software maintenance a project. In a software maintenance project, generally there is a contract between the parties to take care of a specific application for a given period of time, e.g., 1 or 2 years, but a contract can be extended as long as both parties remain satisfied with each other's performance or as long as the application is in commission. An overall contract would specify:

- Billing rates
- Mode of requesting work
- Service level agreement (SLA) specifying the priorities and turnaround times
- Persons authorized to initiate/authorize work requests, accept deliveries, give clarifications
- Escalation mechanisms
- Billing cycles and payment schedules

This list could go on forever, depending on the specific needs and the "pain" of the organizations. Normally, a maintenance work request (MWR) triggers software maintenance work. An MWR can be known by other terms, depending on the organization:

- Program modification request (PMR)
- Program change request (PCR)
- Defect report
- Software change request

Again, this list can also go on forever. Contractually, an MWR is expected to have proper authorizations and to have them in advance. However, for an immediate need, a telephone call, a fax, or an email can also be used and later regularized through raising an MWR, i.e., post-facto (although frowned upon as potentially leading to loss of control).

Work included in a software maintenance project is classified into five types: defect fixing, operational support, fixing odd behavior, software modification, and functional enhancement.

a. Defect repair

Defect fixing work involves fixing a reported defect. A defect may be classified as:

- Critical (a "show-stopper")
- Major (hinders smooth functioning of work)
- Minor (mostly a nuisance; work is not affected)

Typically, defect fixing has an associated SLA in which the turnaround time for each class of defect, based on priority, is defined (i.e., the time between when a defect is reported until the time it is fixed, the regression test is completed, and the software is handed over to production). Sometimes, the turnaround time can be as little as hours or minutes, depending on the application and the needs of the organization. Normally, the maximum turnaround time for fixing a defect

would be about 2 days. In a defect-fixing scenario, follow-up and progress reporting are frequent and close together. Generally, the steps in fixing a defect include:

- Studying the defect report
- Replicating the defect scenario in a development environment
- Studying the code
- Locating the defect
- Fixing the defect, conforming to code change guidelines
- Arranging for peer review and implementing feedback (if any)
- Arranging for independent regression testing and implementing feedback (if any)
- Delivering the fixed code to production for implementation in the production system
- Closing the request

b. Functional expansion

When additional functionality is required in existing software, functional enhancement is the tool to achieve it. Functional enhancement work is generally of longer duration and may range from a calendar week upward. Work included in functional enhancement includes:

- Adding a new screen or report
- Adding additional processing functionality (e.g., quarterly/half yearly/ yearly processing)
- Adding a new module in the software
- Integrating with another software
- Building interfaces with other software
- Adding new hardware and building an interface to the new hardware in the existing software

Functional expansion generally fits the full SDLC model in which the project leverages the full software engineering process and the project management process and can be treated as an independent project if the duration is sufficiently long enough. The level of process rigor required is typically driven by risk. Each organization has a different definition of a project that should be treated as a functional enhancement project. For example, in one organization, a functional enhancement project is defined as "work with the duration of one person-month of effort or more," while in another, the definition of a functional enhancement project is "40 hours of effort."

c. Operational support

Operational support is similar to defect fixing. Many times, operational support requires immediate attention. Activities under operational support include:

- Running periodic jobs (end of day/week/month)
- Taking backups
- Restoring from backups
- User management functionality (including creation, deletion, and suspending of user accounts and changing access privileges, etc.)
- Providing "hand-holding" assistance at a specific workstation
- Extracting data and producing an ad hoc report on an urgent basis
- Providing a temporary patch so that operations may continue
- Investigating operational complaints

Again, the list of activities is long and varied.

d. Fixing odd behavior
In large, complex software systems, and in systems that have been in existence for many years and have undergone software maintenance (e.g., defect fixes, software modifications, and functional expansions), random defects may often crop up under some circumstances, but not in others. These random defects are generally difficult to replicate in a development environment. One reason is because the defect occurs in the field and the person witnessing the defect does not note the chain of events that caused the defect. So until the defect becomes chronic, it might have been handled as an operational support activity and not have been recognized as defect. Such puzzling defects can be placed in the odd behavior category of software maintenance. Odd behavior can be caused by the application software or the system software, a client workstation or a virus, network security, or a combination of all of these. Diagnosing and correcting odd behavior issues may take longer than a week because correcting odd behavior is similar to conducting research. General steps in fixing odd behavior include:

    - Studying the odd behavior report
    - Trying to replicate the behavior scenario in a development environment
    - Studying the code
    - Listing all possible alternative reasons for the reported behavior
    - Reviewing the code for each alternative for possible opportunities for improvement
    - Iterating/eliminating all causes, one by one
    - Fixing all possible opportunities for code improvement
    - Arranging for peer review
    - Arranging for independent regression testing
    - Delivering the software to production for implementation of improved code in the production system
    - Waiting for another report of the identical odd behavior and repeating all the above steps.
    - Keeping the request open through a period of observation

e. Software modification. Software modification work is the bulk of software maintenance in most organizations. Modification of working software is necessitated due to:
    - Changes in requirements mainly due to changed conditions occurring over a period of time
    - Changes in business processing logic
    - Convenience for users
    - Changes in statutory requirements

    Often, modifications include changes to reports, changes to screens by moving around data fields, adding or deleting a data field or two, or some other small enhancement. Steps in the process of software modification include:

    - Studying the software modification request
    - Analyzing the existing software to identify components that require modification

- Preparing a design modification document and obtaining approval from appropriate executives
- Implementing the approved design modification in the code
- Arranging for peer review of the modified code
- Arranging for independent functional testing of the modified functionality — to ensure that it conforms to the approved design document — and implementing feedback (if any)
- Arranging for independent regression testing and implementing feedback (if any)
- Delivering the modified artifact to production for implementation in the production system
- Closing the request

4. Web Application

Web projects refer to Web-based application development projects. Web projects differ from other projects because they have more than two tiers:

- Presentation tier
- Database server tier
- Application server tier
- Web server tier
- Security server

A Web application consists of:

- HTML pages that include graphics to enhance the "look and feel" of the Web pages
- Backend programs for data manipulation
- Middleware programs for application server or rules engines
- Middleware programs for security management
- Other application-specific programs

Another notable feature of Web applications is that backend programming and middleware programming may be in different programming languages and may require persons with different skill sets, even for the same project. Another request is for independence from databases and Web browsers, which necessitates coding routines that are not oriented toward functionality. Additionally, a Web application needs to be developed so that it facilitates an easy change of code. Environmental changes that have nothing to do with the organization, e.g., a new security threat, the release of a new browser, or the upgrade of an existing browser, etc., can also trigger software maintenance in a Web application — even though the functionality remains unaltered. Web-based and client server projects have a very similar profile.

5. Agile Development

Agile software development refers to a group of software development methodologies based on iterative development, in which requirements and solutions evolve through collaboration between

self-organizing, cross-functional teams. (Agile project management is discussed at length in Chapter 11.)

CONCLUSION

Software projects are basically projects with a definite beginning and a definite ending, except that the final end product delivered is not physical. Software projects come in various types and sizes. Product maintenance in software is also treated as a project — unlike physical product maintenance. This chapter defines software projects as well as enumerates the different types of projects, laying a foundation for better assimilation of the science and art of software project management. Subsequent chapters will deal with the subject of software project management, building on this foundation.

2 APPROACHES TO SOFTWARE PROJECT MANAGEMENT

Software project execution has two components, namely, software engineering and management. Software engineering consists of all of the technical activities that are performed to build the project deliverable (the "just build it" activities). Software engineering deals with constructing the components, integrating them, verifying them, validating them, and finally combining all of the components into a product and convincing the customer to accept delivery of it. Management facilitates software engineering so that the project deliverable is completed on time, efficiently and effectively, and without defects.

ALIGNMENT OF SOFTWARE ENGINEERING METHODOLOGY WITH PROJECT MANAGEMENT METHODOLOGY

There are two general schools of thought about the linkage of the software engineering and management methodologies: tightly coupled and loosely coupled.

Tightly coupled. One school of thought maintains that both of the methodologies are tightly coupled and that management is completely dependent on the software engineering methodology adopted for building the project deliverable. Therefore, project management needs to be tightly interlaced with software engineering.

Note: In some software engineering methodologies, such as agile methods, the distinction between software engineering and project management is somewhat blurred. In this situation, the argument is that the SPM or software project manager acts as a coach because the primary responsibility of an SPM is to be the voice of the people and a leader rather than a director.

Loosely coupled. In the other school of thought, the two aspects of software engineering and management are loosely coupled, but they do influence each other. Therefore, each aspect needs some amount of tailoring to suit the other. Additionally, in this school of thought, project management is considered to have multiple objectives, with the primary objective being to build the deliverable. Other objectives include management of the schedule, productivity, quality, resources, morale, customers, and profit. In the loosely coupled school of thought, an SPM is to be a manager first and to be aware of the software engineering methodology second.

Briefly, software engineering methodologies include waterfall, incremental, spiral, object-oriented, use case-based (unified modeling language), and agile methods of various types. These methodologies are also commonly referred to as SDLCs (software development life cycles). Agile methods include extreme programming (XP), scrum, clear case, feature-driven development, test-driven development, dynamic systems development, rational unified process (RUP, the agile version), adaptive software development, and pragmatic programming. Agile methods, with the exception of RUP, encourage only the development of the minimal required documentation associated with the software's development. RUP, however, is an exception because RUP is a detailed software engineering process that includes levels of documentation that are similar to other types of methods.

The alignment of the project management methodology to the software development methodology is driven by a number of factors, such as organizational size and the form of software engineering used on

a particular project. For example, in a small organization in which the owner is a technical person who is actively involved in project activities, the management methodology can completely align with the software engineering methodology. In other cases, e.g., when types of projects and project management styles are more varied, using disparate software engineering methodologies, then aligning the project management methodology completely with the software engineering methodology, is problematic.

A completely aligned project management methodology is suitable for smaller, more homogenous organizations, while less homogenous organizations should have a project management methodology that is decoupled from the software engineering methodology of the project. Because the methodology of software engineering used on a project has an impact on project management, each project will need to have the management methodology tailored to some extent to align with the software engineering methodology.

THE AD HOC METHODS-BASED APPROACH

By definition, ad hoc methods are not documented and are dependent on the involved parties. In an ad hoc method approach, a software project manager (SPM) is given almost absolute control within a general policy framework that tends to be rather flexible. In organizations that allow ad hoc methods, management typically dictates policy and then modifies the policy as necessary or when convenient. In this situation, often the management style also reflects the personality of the leader of the organization. Management driven by the personality of the leader is classically referred to as "hero-driven" management. In organizations with hero-driven management, success is more luck than process supported.

Advantages of the ad hoc methods approach are that it:

- Fits a dynamic environment
- Allows the leader to have absolute control
- Is perceived to allow very fast response to environmental changes
- Can be the least costly, and the most profitable, methodology (with a well-seasoned SPM and if nothing surprising happens)
- Is perfect for pinning the blame for failure on one person (Always have your CV ready!)
- Reduces process overhead activities to nearly zero (e.g., process definition, maintenance groups, measurement, and analysis)
- Permits the principle of "unity of command" to be implemented (can be a great motivator for people involved in a project)
- Leads to a sense of "heroism" in management styles

Disadvantages of the ad hoc methods approach are that it:

- Creates uncertainty in the workplace
- Fosters a leader-centric environment rather than an environment driven by organizational and project goals
- Centralizes authority: lose the leader, lose the project
- Results in outcomes being unpredictable (because they are person-driven)

- Focuses on people monitoring rather than on overall project monitoring
- Causes organizational bandwidth (the capability of the organization to handle multiple projects concurrently) to be dependent on a leader's capacity (to manage multiple projects simultaneously, work long hours, etc.)
- Causes growth in an organization to be limited by the capacity of an SPM
- Leads to deterioration of morale in the workplace due to the encouragement of an undesirable, ego-driven environment (e.g., encourages an increase of self-serving sycophants)
- Hinders (or makes impossible) the development of leaders from within the organization (employees work in their own "cocoons")
- Hastens the inevitability of failure of human endeavors

All in all, the ad hoc method approach might produce some grand successes, but these successes are not sustainable. Because of the inevitability of failure in human endeavors, the impact on the organization can be severe: failure cripples an organization — from which it may not recover. Despite the risks, however, ad hoc approaches to software project management continue to be adopted by a significant number of organizations.

THE PROCESS-DRIVEN APPROACH

Organizations using a process-driven approach are characterized by having documented processes for all activities. Individuals must also be knowledgeable in the processes that concern them to be effective and efficient.

An organization that adopts a process-driven approach to software project management recognizes that the onus is on the organization as well as the individual SPM for ensuring continued project successes. The organization facilitates the execution of projects by providing the processes, the tools, a knowledge repository, the training, and expert assistance as needed to help the SPM(s). In other words, the organization's infrastructure enables successful execution of projects by the SPM. In addition to facilitation, the organization also has the responsibility of project oversight, monitoring, measurement, and benchmarking and effecting improvements in processes, tools, and the knowledge repository to continuously keep the organization well honed.

Each SPM is responsible for executing projects while diligently conforming to defined processes provided by the organization. SPMs also are responsible for providing feedback, suggestions, and support for the organizational initiatives in continuous improvement of processes, tools, and the knowledge repository.

In a process-driven organization, the organization and the SPM work in a close-knit manner, complimenting each other's efforts. The goal of the process-driven approach is to achieve uniformity across the organization in project execution regardless of the SPM involved. An often-mentioned benefit of the process-driven approach is that it enables the free movement of people from project to project with no discernable impact on project execution.

Advantages of a process-driven approach to software project management include:

- Minimizes the person-dependency of project management
- Enables a beginner in project management to perform like an expert and an expert to excel
- Facilitates the "plowing back" of experience gained from project execution into the process (As a result, every project execution enriches the process.)
- Equips everyone with the best practices in the process culled from project execution.
- Monitors projects rather than people
- Involves the organization in project execution and organizational expertise, not only from the process, but also from senior executives whose considerable experience influences project execution and supports its continued success
- Provides uniformity of project execution across the organization, irrespective of the people involved in the project, which leads to organizational maturity
- Facilitates measurement, resulting in fair performance appraisals, which makes possible real morale improvement in an organization (Having measurements also facilitates benchmarking the organizational performance with similar organizations and enables improvement thereof.)
- Builds the basis for predictability in project execution
- Enables all-round participation; iteratively drives an organization toward excellence
- Promotes the recruitment and induction of new people into projects because process-driven processes facilitate raising newcomers' performances to acceptable levels quickly

All in all, a process-driven approach facilitates person-independence in project execution, while facilitating process improvement and moving toward uniformity in project execution across the organization — characteristics that tend to foster organizational excellence.

SO, WHAT IS THE RIGHT APPROACH?

The field of software development is characterized more by diversity than by homogeneity. Therefore, attempting to prescribe one "right" approach is neither feasible nor appropriate. Although this book deals with software project management from the viewpoint of an organizational approach to software project management that utilizes a process-driven approach, we will briefly describe the characteristics of the ad hoc approach.

The Ad Hoc Approach

An ad hoc approach will serve well organizations in which:

- The organization is small.
- The number of SPMs in the organization is small (e.g., two or three). Having a few SPMs facilitates resolution of differences between the methodologies adopted by the SPMs through progress-monitoring meetings. Small organizations do not need to have a set of documented project references. The senior manager can ensure uniformity through personal intervention and act as the resolution mechanism when there are differences of opinion.
- The number of concurrent projects is five or less. Having a small number of projects also makes resolution of differences in project execution easier.

Most organizations that evolve in an organic manner start small, typically beginning with an ad hoc approach to software project management. As the organization grows and takes on more projects, the workload increases, putting pressure on the human resources. As this pressure on the human resources builds up, two things can happen: the organization buckles under the pressure and moves toward stagnation, failure, and closure or the organization moves toward a process-driven approach.

The Process-Driven Approach

When an organization embraces a process-driven approach, it first adopts a process improvement philosophy and the development of processes to cover project management. Then the organization moves on to address additional organizational areas and movement toward a more mature process — which would seem to be a natural progression. However, embracing a process-driven approach proactively when an organization is at the "take-off" stage is better than having to be forced to do so by the complexity created from the sheer volume of work.

But Is a Process-Driven Approach the Right Choice?

Some organizational activities are already process-driven. For example, in almost every country, through statutes, financial accounting is the first activity to adopt a rigorous process, especially in companies or organizations that handle funds from the public. Strict internal controls and external verification through audits are also mandatory. In addition to auditors, other statutory groups act as "watchdogs" over these organizations.

Human resources (HR) departments are often the next area that adopts a process-driven approach. The need for a process-driven approach in HR departments, however, results from the need to ensure fairness to candidates approaching the organization for employment and to ensure that the right human resources are supplied within the organization. Additionally, the unionization of workers as well as other statutes enacted to ensure fair working/hiring conditions increase pressure on organizations to adopt a process-driven approach in their HR departments. In these two examples, each department has an objective of ensuring fairness, but their main objective is to deliver actual results.

In the project environment, however, the aspect of fairness in delivering results is not mandated by any statute. Thus adopting a process-driven approach for reasons of fairness is optional, which leads some organizations to disapprove of a process-driven approach. In these organizations, some even go so far as to say that any process-driven approach is a restriction of their freedom to act creatively. In organizations that do not practice process-driven management, but prefer ad hoc methods of project management, often heard are statements such as "results at any cost" and "by hook or by crook." Other statements that indicate an ad hoc project management approach are hearing a senior manager tell subordinates, "I do not know (or care) how you do it — but I want it by …."

If all of the "heroic" types are curtailed, and a true factory approach replaces a project-oriented environment, when success is realized, all of the stakeholders are heroes, not just a few individuals! Continued success leads to the organization becoming "heroic" as well. As a result, all of the employees perceive that they are wearing the "crown of a hero" — something all organizations striving for excellence pursue. Organizational success belongs to its people. For example, employees of

organizations such as IBM, Microsoft, GE, etc. often feel heroic, while employees at other less successful organizations have "chips on their shoulders" and often are jealous of these heroic organizations.

Conversely, an organization that adopts the ad hoc approach produces heroes by luck or chance, while a process-driven organization makes everyone in the organization a hero/heroine! An organization that adopts an ad hoc approach may survive on the heroics of its employees, but a process-driven organization runs like a well-oiled machine, without the necessity of wide-scale heroics by anyone.

From the beginning, however, software engineers have resisted any move toward adopting a process-driven approach. For example, the waterfall model was the first process-driven approach to software development, but in today's environment, more people continue to hate the waterfall model rather than to love it. As a result, many other approaches, namely, rapid application development (RAD), joint application development (JAD), incremental, and agile methods (e.g., XP, scrum), were developed to replace the waterfall methodology. Although methodologies come and go, and some are forgotten, interestingly the waterfall model is still in the mix.

Even today, strong resistance to the process-driven approach continues and numerous write-ups continue to extol the virtues of ad hoc approaches to project management. So our advocacy of the process-driven approach is fraught with the prospect of stiff resistance. Still, we believe that the process-driven approach is a means to assure project success — the first time and almost every time. We think the process-driven approach ensures organizational success in the short term as well as in the long term.

In a Process-Driven Approach: What Process and How Much?

Once we decide to adopt a process-driven approach, the next questions to address are what type of process should we adopt and how deeply should the process penetrate into organizational functioning?

When deciding about the type of process to use, there are two predominant, popular, process standard frameworks: ISO 9000 (of the International Organization for Standardization) and CMMI (Capability Maturity Model Integration of the SEI, the Software Engineering Institute of Carnegie Melon University). ISO covers the entire organization, while CMMI covers activities that are specifically relevant to software and hardware products. Both frameworks advocate a process-driven approach. The nuts and bolts may differ, but the main premise remains similar. Although, there are other frameworks, these two are predominant.

So, what does a process-driven approach contain? From a simple point of view, a process-driven approach consists of:

- Processes for carrying out the activities
- Agencies responsible for carrying out the activities
- Processes for ensuring that quality is built into the deliverables
- Agencies responsible for assuring quality in the deliverables
- Processes for defining and maintaining organizational processes
- Agencies responsible for defining and maintaining organizational processes
- Processes for measuring and analyzing the process performance
- Agencies responsible for measurement and analysis of process performance

In short, a process-driven approach contains defined methods for carrying out the work as well as the checks and balances necessary to ensure that the processes deliver results and that everyone adheres to the processes.

A simple framework for project management is comprised of project acquisition, initiation, execution, and closure. Once acquired, project processes and project management processes begin with project initiation followed by project execution and project closure. Thus, at the project level, there should be project management processes for:

- Project initiation
  1. Review and revision(s) of preliminary estimates
  2. Identification and acquisition of necessary resources
  3. Finalization of service level agreements (SLAs) between various stakeholders of the project
  4. Preparation of project plans
  5. Conducting induction training for team members
  6. Kickoff of project

- Project execution
  1. Work management
  2. Configuration management
  3. Quality management
  4. Productivity management
  5. Team management
  6. Customer management
  7. Measurement and analysis
  8. Project monitoring
  9. Reporting and escalation
  10. Project delivery
  11. User training
  12. Documentation

- Project closure
  1. Release of project resources
  2. Documentation of best and worst practices as well as lessons learned in the project
  3. Identification of reusable components and documentation of their design and usage
  4. Updating of the skills database
  5. Updating of the knowledge repository with lessons learned and best and worst practices
  6. Updating of the code library with reusable components
  7. Conducting the project postmortem
  8. Conducting a knowledge-sharing session
  9. Release of the software project manager (SPM)

In addition to project acquisition, initiation, execution, and closure, which are the core phases of project management, the organization also has a role in ensuring the success of projects; hence, project management should also have organizational-level processes for:

♦ Project acquisition

1. RFP (request for proposal) scrutiny (a feasibility study in the case of internal projects)
2. Cost estimation
3. Proposal preparation and submission
4. RFP follow up
5. Obtaining the order (obtaining budget approvals in the case of internal projects)

- The PMO (project management office)

1. Project initiation

- Identification of software project manager (SPM)
- Allocation of resources for the project
- Finalization of SLAs (service level agreements) between various stakeholders of the project
- Kickoff of project

2. Project execution

- Project monitoring
- Exception reporting
- Measurement and analysis at the organization level

3. Project closure

- Takeover of project records
- Coordination of knowledge sharing
- Measurement and analysis

    1. Measurement procedures
    2. Analysis procedures
    3. Process capability determination procedures
    4. Metrics reporting procedures

- The training process

    1. Identification of organizational training needs
    2. Fulfilling skill gaps uncovered during training needs analysis
    3. Maintaining a skill database for all organizational human resources
    4. Maintaining a training material repository as part of the organizational knowledge repository
    5. Taking ownership for maintaining the organization at the cutting edge of the organization's chosen area of expertise

- The knowledge repository process

    1. Identifying components of the organizational knowledge repository
    2. Designing, building, and maintaining the organizational repository
    3. Periodically carrying out cleanup of the repository

- Process engineering group processes

    1. Defining and maintaining organizational processes
    2. Defining process and quality audit processes
    3. Defining roles and responsibilities

- Software engineering processes

    1. Requirements processes
    2. Software design processes
    3. Software construction processes
    4. Software testing processes

With the exception of the software engineering processes, all of these processes will be discussed in detail in subsequent chapters.
Briefly, software engineering processes describe the technical side of software development. There are several methodologies for software engineering; however, a detailed description of these methodologies is beyond the scope of this book. In subsequent chapters, however, we will discuss the influence that each of these methodologies exercises on software project management.

3 SOFTWARE PROJECT ACQUISITION

The first activity in software project management is the acquisition of a project. Project acquisition is an activity carried out primarily by a business acquisition team with the assistance of a technical team. (In smaller organizations, project acquisition can be conducted by a single person or a team, depending on the distribution of labor. Key to remember is that the tasks required to seek out and acquire the work must be accomplished by someone — whether by a single person or by a team is immaterial.) Deciding on the project to be undertaken is a very important strategic activity because the decision made will have a huge influence on the organization's financial health and profitability. Project acquisition should therefore be a *collaborative* activity between the business acquisition team, technical team, finance team, and senior management. The focus of senior management is from a strategic perspective.

Two typical project acquisition scenarios are acquisition from an *external client* and acquisition from an *internal source*. Each of these scenarios has a different workflow.

FROM AN EXTERNAL CLIENT

In a scenario in which a project is acquired from a prospective client (an *external* organization), the project is to be a revenue-generating device. Characteristics of a project from a prospective client include:

- The project will result in revenue for the acquiring organization from the external client.
- The software product resulting from execution of the project will be used or resold by the external organization.
- The external organization will impose stipulations on quality, schedule, and cost.
- End users will typically *not* be directly accessible to the software development team. (The development team interacts with the end users via a proxy.)

Project acquisition from a prospective client follows several steps:

1. The request for proposal
2. The proposal
   - Software estimation
   - Delivery commitments
   - Pricing the proposal
   - Preparing the proposal
3. Negotiation
4. Contract acceptance

These steps will now be explained in greater detail. The process of acquiring a project from an external customer is also depicted in Figure 3.1

The Request for Proposal

Usually, project acquisition begins with an RFP (request for proposal), or alternately with an RFI (request for information), from a prospective client. An RFP is obtained from a prospective client by the

organization's business acquisition team. (Because obtaining an RFP is a core marketing activity, obtaining an RFP is therefore not covered in this book.) The RFP document normally contains the following:

- Name of the prospective client and other details about the client
- Name of the project and other details about the project
- Contact details for the project coordinator of the prospective client
- Scope of the work and terms of reference
- Bidding details, including the format of the bid document (either a single-bid system, technical bid and financial bid in *one* document; or a two-bid system, technical bid and financial bid in *two* different documents and submitted separately); any requirements for a bank guarantee/ earnest money deposit toward project execution warranty; etc.
- Procedure for evaluating bids that are received by the prospective client
- Important dates, including the date for requesting clarifications; the date for submission of bid; the date for awarding of the project; etc.

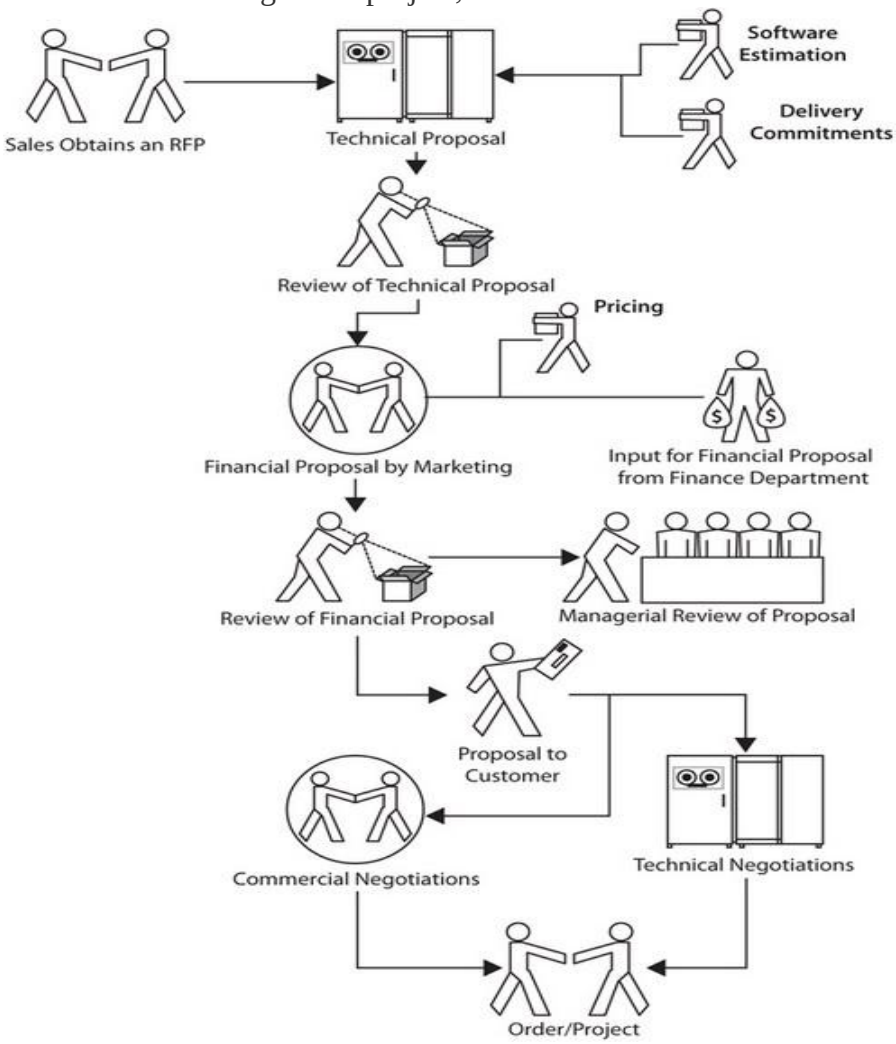

Figure 3.1. Project acquisition from an external customer.

Usually, the business acquisition team initially reviews an RFP to ensure that the information contained in the RFP is complete. Next, the RFP is passed on to the project management office (PMO) or to the

head of the technical team responsible for delivery to clients. Then the RFP is assigned to a software project manager (SPM) for preparation of the technical proposal. To determine the feasibility of execution, the SPM reviews the RFP to ensure that areas describing the scope of work and the terms of reference and suggested technology, if any, are complete. Then the SPM prepares the software estimate for the project. Other components of the estimate (e.g., hardware, business processes, etc.) are handled by the appropriate project managers or management groups.

The Proposal

Arriving at the proposal includes software estimation, delivery commitments, pricing the proposal, and preparing the proposal, which will now be explained in more detail.

Software estimation. Software estimation is carried out at this point to set a price for the proposal. Estimators need to keep in mind that software estimation assists project pricing, but is not the same as project pricing. Project pricing is a commercial decision. (Pricing methods will be described later in this chapter.) Software estimation has four dimensions:

- Estimation of the size of the software product to be produced
- Estimation of the effort in person hours or person days that will be required to develop the specified software product
- Estimation of the cost that will need to be expended to execute the project
- Estimation of the duration (schedule) in calendar days or months that are necessary to execute the project

At this stage, the purpose of software estimation is to:

- Assist in the pricing decision (because the software development effort is a major cost component in the price of the project)
- Estimate the resources required to execute the project (human, machine, money, and duration)
- Assist the decision makers in determining whether the organization has adequate bandwidth in terms of resources to execute the project as detailed in the RFP
- Assist the decision makers in making delivery commitments to the prospective client

Many methods are used to develop an estimate, ranging from parametric models to relational estimates. Usually, the SPM carries out the software estimation based on the organization's software estimation process. The estimates are then subjected to peer and managerial reviews. After implementing the review feedback, if any, and closing the review reports, the SPM submits the software estimate directly to the requestor or makes it a part of the technical proposal. (Software estimation is an independent subject and hence is not covered in this book. A suggested source of additional information on this topic is Software Estimation — Best Practices, Tools and Techniques by Chemuturi.1)

Delivery commitments. Delivery commitments are usually a part of a technical proposal, but as in project pricing, keep in mind that delivery commitments are also a commercial decision. Delivery commitments depend on:

- The prospective client's requirements/urgency
- The competition (which might offer shorter delivery commitments)
- The possibility of a future project that will require the release of resources from this project
- Any other specific reason

Note: The schedule should not be influenced by the above considerations. If a shorter schedule is committed to by the decision makers, explore ways to meet the commitment: either by the infusion of more resources or more expert resources, or by subcontracting a part of the project, or by the project team taking on more stress.

Pricing the proposal. Pricing is a complex interaction of various factors, including commercial considerations, the perceived necessity/desirability of acquiring the project, the opportunity at hand, and pricing models — considerations which dictate pricing for a project. Pricing strategies include:

Order book position: If an organization's order book is empty (no ongoing projects at the present or none in the near future), the organization might price a project low, so that at least the project will pay the organization's variable costs. Conversely, if the organization's order book is full (at the present as well as in the near future) and it does not wish to expand its capacity, the organization might price a project high.

Necessity to obtain an order: Sometimes winning a particular order is critical for reasons other than just the need to fill the order queue. In this situation, an organization might price a project just low enough to win the bid. Reasons for this scenario include:

- The project will open up avenues of future business.
- The project will provide significant visibility.
- The project will generate significant free publicity.
- The project will provide an opportunity to train a number of the organization's resources (personnel) in a cutting-edge technology, from which the organization can win future orders at higher margins.
- The project will result in the organization gaining a foothold in a new and lucrative market.

Desirability of winning the order: Sometimes an organization does not desire to obtain the order, but must pursue a particular client's business for "political" reasons. In this situation, one strategy is to price the project so high that the organization is certain not to win the bid. Reasons for this scenario include:

- The organization does not have the technology to execute the order.
- The organization does not wish to do business with the client (perhaps the client is slow to pay or otherwise has a bad reputation), but there is a strong business reason to answer the RFP.
- The project involves obsolete technology or technology that is becoming obsolete, causing the organization's resources to be disinterested in the project or unwilling to work on it.

Stiff competition in the field: The organization might price a project with a bare minimum profit margin just to compete with other organizations.

a. Monopolistic:

Sometimes an organization is in the unique position of being the only supplier of a product or service. In this situation, the organization exploits (skims) the opportunity and gets as much revenue from the order as possible. In this situation, pricing is focused on maximizing the return from the project.

b. Oligopolistic:
An organization might collaborate with other vendors in the market to establish one price for a product or service that is profitable for all of the vendors involved. (The authors note that this strategy is of questionable legality, especially in North America and Europe, but oligopolistic activity does occur; therefore, it is included for completeness.)

c. Repeat orders:
The organization offers a service to an existing customer at a price that is fair to both the organization and the customer.

d. New market opportunity:
If an order facilitates entry into a new market — either geographical- or domain-based — the organization will establish pricing in a manner that maximizes its chances of getting the order.

Let's now look at the pricing models that are applicable to software projects. Pricing models are a means of implementing a pricing strategy. Any strategy can leverage one or more pricing models. Some popular pricing models that are available to free-to-price organizations (i.e., organizations not constrained by a government statute when setting prices or situations in which market forces set pricing) include:

a. Time and material pricing:
Time and material (T&M) pricing involves agreeing on an hourly rate for each category of human resources engaged in the project and then charging for these resources based on the actual time spent on the project. Time spent is assessed using time sheets that have been approved by the client. (T&M pricing also includes other expenses. For example, travel expenses incurred by human resources on activities that have been approved by the client are charged at the actual cost incurred by the human resources.) Although economic pressure appears to be moving more contracts to a fixed-price model, the T&M model is still in existence.

b. Cost plus: Cost plus is a typical model in project pricing. In cost plus pricing, an organization accounts for all of the costs and then adds a reasonable profit to the final cost. The costs of the vendor are transparent to the purchaser. The cost plus pricing model is commonly adopted between partners who have a strong, long-standing bond. For the most part, cost plus pricing is a fixed-price model, sometimes with a provision for cost escalation for changes that are requested and approved by the client.

c. Opportunity: In the opportunity pricing model, pricing depends on the opportunity presented by a potential purchaser. If the purchaser has no choice but to buy from a single selling organization, the organization can charge top dollar. Conversely, if there is too much competition, the organization might charge a lower price. The organization might even price a project at a loss if a prestigious project would provide significant publicity to the organization. Within the opportunity pricing model are two variants:

- Penetration: A new entrant into a market adopts penetration pricing in a market that is already overflowing with existing providers. To gain a foothold in such a market, a new entrant charges lower prices as an incentive to motivate purchasers to entrust the project to the new entrant. Sometimes the penetration pricing model is known as introductory pricing.
- Skimming: Skimming or high-margin pricing is used by an organization that has an advantage, such as being an "early bird" in an emerging market. The organization charges higher prices before their competition enters the market; then, once competition is firmly established, the organization lowers prices.

d. Going rate:
The going rate pricing model is adopted by organizations in fields in which there is abundant competition and the price for a particular product is well known to purchasers. The going rate pricing model entails pricing a product (or service) at a price that is similar to the competition. (T&M pricing is typically dictated by the going rate for the cost of human resources.)

e. Monopolistic:
Monopolistic pricing is a variant of going rate pricing. In the monopolistic model, the seller promotes a unique feature of its product and thus prices the product higher or lower than the going rate. Monopolistic pricing is frequently used when a two-bid system (technical bid and financial bid) is adopted by a purchaser.

f. Oligopolistic:
The oligopolistic pricing model is used in a market that has a limited number of suppliers. Suppliers collaborate (sometimes by forming an association) to establish a fixed price for a product and then to enforce that price — the price is the same from all suppliers. (Note: This strategy is of questionable legality, especially in North America and Europe, but oligopolistic pricing activity does occur and therefore is included for completeness.)

g. Transfer:
The transfer pricing model is adopted by two departments that are within the same organization. In this scenario, only the actual cost is transferred from one department to another. Transfer pricing is also known as a chargeback model.

h. Loss leader:
Loss leader pricing is adopted by organizations that are trying to lure clients away from their current vendors. In the loss leader pricing model, an organization provides a product (or service) at a loss to a customer, speculating that as a result of the pricing strategy, increased additional business from this customer will offset the losses. Alternatively, an organization may offer free publicity or a prestigious status to new buyers, which might result in new business from other potential buyers.

Because an RFP presents a project opportunity for an organization, a business acquisition team considers the opportunity from a business perspective and then decides how best to utilize the opportunity.

Remember from an earlier discussion that pricing and estimation are two separate scenarios that are generally handled by two different areas in an organization.

Although every organization has its own pattern of implementation, project pricing typically follows a stepwise process:

1. The technical team prepares the software estimates (software size, software development effort, and software development schedule) and presents them to the business acquisition team.
2. The business acquisition team prepares the cost estimation and presents it to the finance team.
3. The finance team suggests a floor price to the marketing team, below which the project will not be attractive financially.
4. The business acquisition team coordinates pricing with senior management (in some cases there might be a pricing committee) and determines:
    - The price to be offered to the potential client
    - The negotiation margin (if negotiations are foreseen)

The price for a project proposal, therefore, is set by an organization in a collaborative, step-by-step manner (with, of course, organization-specific changes in the methodology). Larger organizations have a higher degree of formality, with committees, meetings, and approvals, whereas smaller organizations have a lower degree of formality, engaging in consultations with the concerned agencies. Although the steps listed above are a simplification of the process used in large organizations, by and large they are the major steps taken to determine the project price.

Preparing the proposal

The business acquisition team uses the approved price and then prepares the proposal in the required format (either the format of the organization or the format specified by the prospective client), arranges for peer review and managerial review, and obtains formal approval for the proposal document. The approved proposal document is then submitted to the prospective client. The business acquisition team follows up with the prospective client until either the project is acquired or is lost to a competitor.

A proposal is usually a multipage document that has two separate parts (even if both documents are submitted in the same envelope): the technical proposal and the financial proposal.

A technical proposal

A typical technical proposal usually contains the following sections:

- Title page: Include a title page containing the name of the project, revision history, the date of the proposal, and the prospective client's name.
- Contents: List the contents of the proposal.
- Introduction: Include information about the organization submitting the proposal and the context of the proposal.
- Scope of work: Include a detailed scope of the work as given in the RFP; any subsequent clarifications issued by the prospective client; and any additional information that the organization may wish to provide.
- Approach and methodology: Describe the proposed execution of the project; the methodology to be adopted for executing the project; and the project management methodology.
- Deliverables: List the deliverables to the client that are expected from the project.

- Approvals required from the client: List the artifacts that need to be approved by the client during project execution, including the timeline required for approvals.
- Schedule for project execution: Depending on specifications from the prospective client or the organization's standards, a schedule can range from a detailed structure that mimics the project to a list of major milestones with dates (or the number of calendar days from the commencement of the project).
- Software estimation: When mandated by the client, include the software size, development effort, development cost, and a detailed schedule. (Software estimates are not submitted with a technical proposal unless they are mandated by an RFP.)
- Inclusions: Describe all of the software engineering activities that are proposed to be performed to derive the specified deliverable.
- Exclusions: List all of the software engineering activities not proposed to be performed during the project execution. Also list any software components that are not being included in the deliverables.
- Responsibilities: List responsibilities of the vendor and the vendee for all major activities proposed to be carried out during project execution. The responsibility for each activity can be a primary responsibility for one party and a secondary responsibility for the other party or responsibility can be primary for both parties.

A financial proposal

A typical financial proposal usually contains the following sections:

- Fee: State the fee for the project.
- Fee exclusions: List items that are not included in the proposal and items that are not included in the project fee (items such as travel, master data file creation, data migration, pilot runs, etc. that may not have been included in the proposal or the fee for the project).
- Validity period: State the period during which the fee offered will be maintained.
- Payment terms: Include any advance fees required; interim payment schedules that are based on period/milestone /delivery requirements; performance guarantees or retention holdbacks; and any penalties for delayed payments.
- Intellectual property rights (IPR): For any deliverables, specify the IPR restrictions on all parties, including the software produced. IPR may rest with the client or with the developer. If a third party component is used, the specific IPR may rest with the party from which the component is procured. All of these aspects should be described in the proposal.
- Force majeure clause: Specify remedies for extreme conditions should they become a reality during the period of project execution (general strife, war, floods, earthquakes, etc.)
- Software tools or components to be supplied by the client: A project may require the use of specialized software tools, components, or system software. If the client is expected to supply such software or components, specify this requirement.
- Facilities: Should the vendor's staff need to work at the client's site, describe the logistics requirements for the workplace that are to be provided for the vendor's staff at the client's site. Specify what organization will be responsible for expenses that are incurred for reasons related to logistics.
- Price escalation clause: Costs may increase, especially for long-duration projects or software maintenance projects. Include an escalation clause that sets out the conditions on which a price

escalation would be based; when the price escalation would be effected; and the mechanisms that would be used to fairly make a decision about a proposed price escalation.

- Arbitration and jurisdiction: Describe dispute resolution mechanisms so that if and when a dispute arises, all parties will understand how a dispute will be resolved, including the arbitration and legal processes. The arbitration/jurisdiction clause should include the conditions under which recourse to arbitration/legal actions may be taken. Additionally, arbitration/jurisdiction clauses typically include names of qualified arbitrators who may be approached to resolve a dispute and the courts of law that have jurisdiction to hear and award judgments.
- Consequential liability: Define the limits of the vendor's consequential liability (sometimes called special damages) should such liability become applicable.
- Other: Include any other items that are relevant to the financial aspects of the proposal.

Negotiation

In negotiating, the client's finance team, assisted by the client's technical team, conducts price negotiations with the vendor's business acquisition team. As in most negotiation situations, give-and-take occurs. For example, the vendor might offer to give some type of discount (price, scope, technical changes, etc.) or the client might place an order after making amendments that would include changes in scope and duration. (Decisions about the maximum amount of discount that can be offered, the minimum duration in which the project can be completed, etc. should be made by the vendor prior to conducting negotiations.) During the negotiation process, decisions are made "across the table" to clinch the deal.

Three familiar bidding scenarios are public bidding, private bidding, and a synthesis of the two. Each scenario has a different perspective:

Public bidding: In a public bidding scenario, a prospective client posts an RFP on the Internet or in the press and invites bids from all organizations that meet the criteria contained in the RFP.

Private biding: In a private bidding scenario, two options are possible:

•An RFP is raised on multiple prequalified vendors.

•An RFP is raised on only one prequalified vendor. (This scenario occurs when an established relationship exists between a client and a vendor.)

Synthesis biding: In a synthesis bidding scenario, open solicitation is combined with the use of a strong core of favored/preferred vendors.

Public bidding

In a public bidding scenario, a two-bid system is usually used (a financial bid and a technical bid). Bids are evaluated based on the criteria specified in the RFP. Proposals are taken as "final." Usually no negotiations are conducted for the purposes of price or technical bargaining. (An exception in price bargaining would be in instances in which a client's budget has been overshot by all bidders. An exception in technical negotiations would be in situations in which no bidder has offered a complete technical solution.) In a public bidding scenario, technical proposals from all of the vendors are evaluated first to shortlist those vendors that fulfill all of the technical requirements and provide the best technical

solution. Then, the financial bids from only the shortlisted vendors are considered. Usually, the lowest financial bid is selected from among the shortlisted vendors.

Private bidding

In a private bidding scenario, especially when raised on only one vendor, price negotiations are usually applied (whether for actual cost or functionality). In a private bidding scenario, because there is an existing relationship, more transparency is expected from each party (less is expected, however, from an external client). In multiple-bid scenarios, even though all of the vendors are prequalified, price negotiations are applied to get the lowest price or to get what is perceived to be the highest quality product. Sometimes the client expects a discount in return for their continued trust in the vendor. Therefore, arriving at a price, including the software size and effort, is derived through negotiation. The private bidding type of pricing tends to cause a business acquisition team to build a "buffer" into the price so that a discount expected by the client can be offered, but planned profitability is still retained.

Synthesis biding.

No typical scenario exists for synthesis bidding. The scenario is rather ad hoc in nature. Synthesis bidding is similar to a private bidding scenario, but to obtain the best technical solution at the lowest possible price, new vendors are brought in to put pressure on existing vendors. The final winner of the project order is usually an existing vendor, although awarding a project in favor of a new vendor is not ruled out.

When negotiations have been successfully completed in an external project, the client places the order to execute the project with the selected vendor.

Contract Acceptance

Once the order to execute a project has been received from the client, and upon receipt of the order, the business acquisition team reviews the order to ensure:

- Price, delivery date, and payment terms are in agreement with those specified in the order.
- All terms and conditions agreed upon and those specified in the order are in agreement.
- Any new conditions are inserted in the order.
- The scope of work is as agreed.

If new concerns are raised, the business acquisition team may need to renegotiate with the client to resolve all contentious issues. In some cases, an order acknowledgment letter is given to the client as a final means of completing the deal. In disputes, the RFP, the proposal, the order, and the order acknowledgment are crucial legal documents. Hence, exercise extreme care when preparing these documents.

FROM AN INTERNAL CLIENT

In this section, internal project refers to two scenarios:

- An organization whose main business is not software development: The organization desires to computerize operations of one of its functional departments. The in-house software development department is to develop the necessary software.

- An organization whose main business is software development: The organization desires to computerize operations in one of its departments. The in-house software development team is to develop the necessary software.

In both of these scenarios, the common factor is that the software is intended for internal use within the same organization. The software might even be used by the software development team itself. The characteristics of an internal project include:

- The resultant software product is not for delivery to an outside organization.
- No direct revenue for the organization will result from this project.
- Project expenditures are treated as a cost to the organization.
- Expected benefits are reduced cost of operations, improved quality, reduced turnaround times, etc.
- End users of the developed software are within the same organization and therefore are accessible to the software development team.
- The project commences with a feasibility study and ends when the software is installed and "goes live" for use by the end users or the project is cancelled. (These types of projects typically go from a development mode to software maintenance mode.)

Expenditures for development of the software and computerization are normally (depending on the cost and applicable rules for capitalization in the locale) treated as capital expenditures and go through the organizational process for sanctioning a capital expenditure budget. Approval of a budget and allocation of funds for the project are usually considered to be sufficient permission for commencement of a project. Steps in acquiring a project from an internal source include:

1. Conducting the feasibility study
2. Preparing the proposal
    - Software estimation
    - Delivery commitments
    - Proposal preparation
3. Finalizing the proposal (from discussions with the end user department or other budgetary groups)

Each of these steps is now described in greater detail.

The Feasibility Study

A feasibility study is usually conducted by business analysts or systems analysts from the software development team. The analysts study the existing system, the documents, and the current process being used.

User requirements. In a feasibility study, the goal of the analysts is to elicit the user requirements for the new software product from the designated end users of the proposed system. Based on the user requirements, the study ascertains the technical feasibility of executing the project.

Technology requirements. The feasibility study determines the technology to be used, including the databases, the software development platform (such as the programming language, software development tools, Web server, application server, etc.), and the hardware and system software requirements. These requirements determine if any new hardware and system software needs to be procured.

Software development approach. The feasibility study determines the possible software development approach. For example, based on the user requirements, can a COTS (commercial off-the-shelf) product with customization be used or does the software need development from "scratch?"

Type of execution. The feasibility study specifies whether the project can be executed in-house or if it should be outsourced. Based on the requirements, if outsourcing is required, the analysts specify the extent/portion of work to be outsourced. For example, in some cases, a part of the software may need to be outsourced; yet at other times, the entire software needs to be outsourced. Creation of the master data files using data entry may also need to be outsourced.

Tangible and intangible benefits. The feasibility study also determines tangible and intangible benefits expected to accrue from the proposed project.

At this point, typically a "ballpark" estimate of the overall cost of the project has also been made. The estimate is comprised of:

- Cost of hardware and system software
- Cost of software development
- Cost of creating master data files or cost of data migration
- Cost of training resources in the new system and cost of change-management activities (activities that are necessary to change operations from the existing system to the new system)
- Any other relevant costs

A feasibility report containing the results of the feasibility study is prepared. A feasibility report usually contains the following sections:

Title page: Provide revision and approval history.

Contents: List the contents of the feasibility report.

Project preliminaries: Provide the name of the project, departments affected, cost center, contact persons, etc.

The project: Describe the project.

Probable benefits: Describe the probable benefits expected from the project as well as the possible negative impacts of not implementing the project.

Cost estimates: Provide the ballpark cost estimates for the project.

Proposed technology: Describe the proposed technology and the reasons for selecting the technology over other competing technologies.

Implementation strategy: Describe the project implementation strategy, including the need for outsourcing (if any), the amount of outsourcing, and the expected duration.

Appendices: Include the following:

- User requirements document
- List of persons interviewed for eliciting requirements

- List of documents referenced
- Details of estimates
- Analyses for arriving at probable benefits, technology proposed, etc.
- Any other relevant material

The feasibility report is submitted to management (or to a capital expenditure approval committee) for consideration and for allocation of budget funds. The approving authority, be it senior management or the capital expenditure committee, considers all competing proposals for available funds for capital expenditure. Capital budgeting techniques such as return on investment (ROI), net present value, and internal rate of return, etc. are used to select and prioritize competing projects. The approving authority then grants (or denies) approval for the project based on the availability of allocable funds and the strategic needs of the organization. If approval for the project is received, the next step in an internal project is preparing the proposal.

Note: Prepare the proposal in line with the approved budget. Budget approval can be granted in full for an entire project as detailed in the feasibility report or for only a part of the project. When budget approval includes only the first few phases of a project, approval for the next phases is not considered until after completion of the sanctioned work.

Preparing the Proposal

Proposal preparation for an internal project is comprised of the software estimation and the delivery commitments for the sanctioned activities as contained in the revised budget. These activities are described in the section on acquisition of an external project; hence, the description of these activities is not repeated here.

The proposal for an internal project is subjected to a peer review and a managerial review and is then submitted for approval by the client — in this case, the head of the user department. For the organization, the proposal forms the basis for carrying out the work and making deliveries of the software product to the end user department. For the end user department, the proposal is used to plan activities to support execution of the project as well as to conduct the follow-up activities needed to effectively make use of the delivered software product.

Finalizing the Proposal

The proposal is discussed with the internal client to ensure that all of their requirements as approved in the budget are met and that the proposed expenditure and timelines meet their expectations. Any feedback received is implemented in the proposal. Once the proposal is approved by the internal client, the project is ready for execution.

Note: In internal projects, the emphasis is not on price, but rather on functionality, expenditures, and meeting required timelines. More, actually maximum, transparency is expected and demanded because both parties are from the same organization. Closer scrutiny of project details is also likely.

Some closing words. Whether a project is from an external client or from an internal source, project acquisition is a preliminary, prerequisite step in project execution. Once the project is acquired, the next

step in project execution is project initiation. The next seven chapters will describe the initiation, planning, execution, execution control, change management, scheduling, and project closure phases of software project development.