# Appendix D: *Templates*

The following project templates can be used as a starting point for any project. Based on the template selected, various phases, activities, and tasks will be created with the correct IDs and customary dependencies. Each task will have the appropriate generic resource(s) assigned. Additionally, each template contains the standard *IT Project Management Phase*, as well as standard contingency tasks and milestones for each phase. Any additional tasks or milestones that are required by the *IT Project Management Standards and Guidelines* are likewise included in each template.

Each template has a small, medium, large, and project phases version. It is recommended that the "*small*" version be selected when creating a detailed project plan. The "*medium*" and "*large*" versions tend to be too large and too in-depth for the typical *IT* project. The "*project phases*" version is most useful for creating a size estimate. Refer to *Section 11: Master Planning, Sizing* for more information regarding size estimates.

G *It is recommended that the "small" version of a template be selected for creating a detailed project plan.*

G *It is recommended that the "project phases" version of a template be selected for creating a size estimate*

A template should be used as the staring point for creating the project's work breakdown structure (WBS). However, every project is unique and it is expected that the project manager will need to make many modifications to the WBS generated by the template. Refer to *Section 2: Planning, Work Breakdown Structure* for more information regarding the standards and guidelines that must be followed when customizing the WBS.

## Client/Server

This guide is intended to aid the reader in understanding the *Client/Server* (C/S) models provided by the project template. What follows are short descriptions of each of the phases and the intent a developer should have in looking at the activities and tasks in each phase. This approach to C/S applications holds several key objectives for success in IT development organizations of all sizes. The following points are key to success in C/S development.

### Enterprise Approach

While an overall architecture is important for all systems development, the nature of C/S approaches and technology requires that particular attention be paid to architectural constructs. C/S development, while effective for single applications, is even more effective when approached with the enterprise in mind. The C/S template includes an *Architecture* phase to realize this concept. Projects can be created using the appropriate planning and implementation phases to build an architecture for the enterprise as a whole.

### Iterative Prototyping

Prototyping is the predominant technique for developing C/S applications. Models are constructed to facilitate iteration of specific phases. For example, the *Prototype Requirements*, *Logical Design*, and *Physical Design* phases are easily iterated when building a project model. Supporting the iterative prototyping, the scope control and management via change control steps have been highlighted in order to ease the expansion of scope and function that naturally occurs when using prototyping techniques.

## Non-prescriptive Models

This is a non-prescriptive method that aids the user in determining what steps to follow in any project. It provides illustrations to develop project models, from which project plans can then be developed with any project management tool. Since most IT development organizations already have standards in place, the C/S template does not prescribe how a particular task should be accomplished. Rather, it allows users to maintain their own standards and forms, in concert with the guidelines provided.

## Phase Objectives

The following table pairs each phase with its primary objective.

| Phase | Objective |
|---|---|
| Business Opportunity | Identify ideas and concepts |
| Project Opportunity | Evaluate project viability |
| Prototyping Requirements | Get the details |
| Architecture | Lay the foundation |
| Logical Design | Synthesize information into a preliminary design |
| Physical Design | Constrain design to fit the technical limits |
| Build | Construct the system |
| Implement | Release to users |

## Architecture

Unique features of C/S applications can cause particular problems on various platforms. The C/S template includes the *Architecture* phase to help development organizations plan C/S applications for appropriate platforms.

Each of the architectures must be evaluated whenever a change to one of them is proposed or implemented. The technical architecture is the primary cause for revision, as the technology changes so rapidly in the C/S environment. However, any number of other factors - for example, changes to a client's expectations in computing - can require modifications in the other architectures as well.

The implementation of guidelines and standards, standardization of hardware and software, and creation of a support staff are worthwhile expenditures that may save time and resources in the future, while decreasing development staff effort in constructing workable systems. If an architecture is in place, application teams are free to focus on the business problems to solve, rather than on the technical approaches to business solutions.

The *Architecture* phase may be constructed as a regular project (using the appropriate *Plan* and *Deploy* stages). However, the architecture will continue to evolve, and the support team responsible for the architecture should be included in each C/S project.

## Enhancement

*Enhancements* include any modifications to an operational system, either to expand its current capabilities or to satisfy changed business, technical, or management requirements. Typically, enhancement projects are of shorter duration than most new system development projects. For this reason, they are considered less complex. However, this may not always be the case. It has generally been proven over time that "*simple*" enhancements or changes are those which result in the most technical maintenance effort. No modification to an existing system should be regarded as trivial, although it may involve less risk of failure than a new system development effort.

Each enhancement project should be managed with the same rigorous application of the methodology as any equivalent-sized new systems development project. Formal plans, schedules, budgets, and sign-offs must be strictly adhered to in order to assure accurate communication of the project's scope, objectives, and constraints, and to provide an effective status reporting mechanism.

Adding an entire new subsystem to an existing system will involve more complexity (new subsystem design, interface requirements, shareable logic) than would the addition of a report or inquiry involving data elements on an existing database. Therefore, selection of tasks and activities from the methodology for enhancement projects is most strongly influenced by the project type and project size.

## Maintenance

*Maintenance* consists of changes made to software to fix errors in code, or to effect operational changes, such as: user procedures, production schedules, file retention procedures, job instructions, or the inclusion of new variables, such as new departments, cost codes, account numbers or validation criteria. It is usually performed "*on demand*."

A maintenance project is concerned with the repair of a "*broken*" application system or function. This differs greatly from those projects that could be undertaken with new development or enhancements. Nevertheless, restoring application services can be viewed as a methodical series of activities and tasks.

Particular note should be made during the technical or production design review phase of the potential impact that the modification will have on the existing system and its files and databases. Pay attention to maintenance modifications which impact input and output activities, since these could also bring about requirements to modify user procedures, functional documentation, file backup and recovery, and data security measures.

Since maintenance activities may range anywhere from simple report heading alignments to processing logic flow, it is extremely important that the individual implementing the change use judgment about its effect. The problem and the solution must still be assessed, planned, executed, tested and documented.

**Note:** Maintenance projects are likely to include direct interface or influence on existing systems functions, files, or databases, whether or not a functional change was intended. Strict and rigorous testing is required. It is highly recommended that the installation and testing activities include a test of the entire system where the number or complexity of effected files or routines makes it feasible to do so.

## New Development

*New Development* is comprised of totally new systems work, or work done on a totally new system. Development is normally handled as a discrete project, in one of the following ways:

- Construction of a new system based on a justified business need.
- Replacement of a manual system by an automated system.
- Major extension of the capability of an existing system.

## Package Installation

*Package Installation* involves examining the feasibility of an applications package to replace existing in-house applications software or manual systems.

Projects of any size that involve the identification, selection, installation, and implementation of software packages have many of the same characteristics of new development or enhancement projects. Special emphasis is required at the beginning of the project to establish a very clear definition of the business or functional requirements. This definition must be of sufficient detail to recognize both the system criteria necessary to satisfy the fundamental "*standard*" business characteristics of the firm and also the unique, business-specific criteria. It is not likely that any vendor's package will completely satisfy all of a given company's requirements.

If too many specifications are imposed (so that all of the criteria must be met), many otherwise qualified vendors may be disqualified unnecessarily. Therefore, a compromise must be considered. If a given package meets about eighty percent of the system's requirements, it should be considered as a candidate for further evaluation.

During the *Feasibility Study* and *Business Area Requirements* phases, an additional set of tasks and activities must be included which will provide for the evaluation of the vendors' companies as well as their products. If a continuing relationship is anticipated, it is best to deal with strong firms that can demonstrate a history of successful implementation of products and a firm financial basis.

Business references, particularly user groups, will help a great deal in this process. This is not to say that small software vendors with relatively new products should be disqualified. They may provide a fit to the functional requirements or provide the most innovative technical approach. Rigorous quality and acceptance testing is required in any case. A high-quality functional and technical education and training program, a proven customer support capability, and a clear understanding of the limits of the software product will make the purchase of the package a success.

The prospective project managers involved in a software package installation must keep in mind that the purchase price of the product and its attendant services typically accounts for only 25 - 50% of the cost of the project. The risk, and therefore cost, during the design and development phases are significantly reduced by a good software package. Its technical risk should be minimal. Therefore, the study, definition, and implementation phases are where most project emphasis is required. Finally, as in other effective projects, it cannot be stated too strongly that user participation is paramount.

A unique characteristic of software installation projects is the change in the relationship between the *Business Area Requirements* (BAR) phase and the *Business System Design* (BSD) phase. In a typical systems development project, the specifications developed in the BAR are carried forward into the BSD to be expanded on or elaborated with additional functional and technical detail. While this may happen if there are any changes to the functional specification of the software package, the BSD phase becomes one more of evaluation of the package than of design. If the package can be accepted and used "*as-is*," little or no *Technical Design* or *Programming* activity may be required.

*Acceptance Testing* must still be planned, scheduled, and carried out in order to validate that the system operates in the new environment and meets the acceptance criteria established by the user.

## Prototype

*Prototyping* for requirements is a system-building approach used to expedite the determination of business need during the development process. It provides a model of the proposed system which is a facsimile in function and a miniature in size. The full-scale system is subsequently constructed using pre-specification system building methods.

The purpose of prototyping is to enhance the ability of users and technicians to define, design, specify, and build effective business system applications, thereby providing the organization with a competitive edge. Development methodologies must continue to incorporate new and better tools to guide the user and technician toward this goal. Prototyping is an approach that can be substituted for traditional specification methodologies during the *Business Area Requirements* (BAR) and *Business System Design* (BSD) phases of the development cycle. The prototype is normally used as the basis for building the full-scale system. When appropriate, prototyping can be used to supplement the traditional specification process.

The traditional approach to the specification of a system using development methodologies calls for the preparation of documentation for the business or logical design (namely, BAR and BSD) and the technical or physical design (including program design). This approach is also known as *pre-specification*; that is, specifying the designs prior to constructing and implementing the business application system. By contrast, the prototyping approach allows system builders to model the proposed system through the development and testing of a miniature logical and physical version of the full-scale system. The miniature version is prepared and agreed upon before the full-scale logical and physical specifications are prepared.

## Team-Specific Templates

There are several templates available that were created specifically for a particular *IT* team: the *Data Warehouse* template, the *Land New Development* template, and the *Web Development* template. These templates were created by *PMO* at the request of the team, and are based on real projects plans that were previously used by those teams.

## Mini-Project

The *Mini-Project* WBS is intended to be used on small projects, less than 200 hours. The tasks are comprised of what would normally be considered phases of a major project, as well as standard *Project Management* and *Contingency* tasks. Any project that is estimated to be over 200 hours cannot use the mini-project template.

S *Any project greater than 200 hours must have a detailed project plan with a Project Management Phase. Projects less than 200 hours may use the Mini-Project plan template.*

## Blank WBS

The *Blank WBS* template creates exactly that – a blank WBS, with no phases, activities, or tasks. This template is typically used by *PMO* to create special projects, such as bi-annual *Enhancement Budgets*. *Project Managers* may want to use the *Blank WBS* template for other out of the ordinary situations. For example, a “*project*” whose sole purpose is to gather related documents in the *Clarity Document Manager* could be created using this template. It is not recommended that the *Blank WBS* template be used for creating a detailed project plan.