

On-Target 4D

Software Development Methodology

By
Craig D. Wilson

Last Modified:
January 15, 2004

Copyright 2001-2004
Matincor, Inc
620 Camino de los Mares, Ste. E350
San Clemente, California 92673-2853

On-Target 4D

Software Development Methodology

Table of Contents

I. Introduction	4
Overview	4
Scope	4
Guiding Principals	5
The Process	7
Team Structure	9
II. Project Management.....	13
Project Planning.....	13
Estimating and Tracking Costs.....	15
Status Reporting.....	16
Project Issues	17
Risk Management	18
Negotiating Internal Resources.....	20
Tracking Baselined Artifacts	20
Managing Outsourced Activities	20
Project Notebook	20
III. Analysis.....	22
Role and Responsibilities.....	22
Scope Document	23
Requirements	24
Conceptual Design	25
Release Notes.....	28
User Guide and Online Help.....	28
Client Representation.....	28
Expectation Management.....	28
Product Management	29
IV. Development.....	30
Role and Responsibilities.....	30
Discovery Phase.....	31
Design Phase.....	31
Use Case Drill-Down.....	32
Design Documentation.....	34
Architecture Review	34
Development	34
Work Package	35
Coding & Unit Testing.....	35
V. Quality Assurance	38
Role and Responsibilities.....	38
Test Planning	39
Test Cases	40

Test Execution	40
Test Result Reporting	42
Testing Environment.....	43
VI Configuration Management.....	44
Role and Responsibilities.....	44
Version Control.....	44
Change Management	45
System Promotion & Build Management	47
VII. Infrastructure	49
Role and Responsibilities.....	49
Physical Infrastructure Design	50
Product Promotion	50
System Configuration Documentation.....	51

I. Introduction

Overview

The On-Target 4D™ software development methodology was developed to provide a pragmatic, easy-to-follow method for guiding project teams in the successful delivery of quality software. The method is based upon current industry practices in object-oriented analysis and design (OOA/D) techniques but the emphasis is on the process. There are a great many works available that discuss OOA/D theory¹ and the practical application of methods to do object oriented design. The focus of the On-Target 4D™ methodology is instead on team roles and responsibilities during the process. OOA/D explains *how* to design an application; this work focuses on *who* does *what* and *when* it should be done. The processes and templates provided in these pages are free for you to use for non-commercial purposes. Matincor, Inc. retains all commercial rights to these artifacts including documents, models, and presentations.

If you find these pages of interest or if you take issue with what I've said, I would like to hear from you. The development of the On-Target 4D™ methodology is on-going. With every project I undertake, I learn something new. My friends and colleagues in the IT industry have helped me to refine my view of software development and I continue to value their feedback.

Scope

There are many well defined software development methodologies available on the market today. Some are “heavy”; that is they provide extensive tools and techniques. One of the most popular is the Rational Unified Process® from Rational Corporation. There are also “light” methodologies such as the more recent Extreme Programming (XP) approach. The heavy methodologies give you everything you will ever need. Unfortunately, they come with a lot of baggage which you may never need and will likely get in the way. (And oh, by the way, they cost a bundle!) The light methodologies tend to be too focused on a limited part of the development lifecycle or are for very small teams. Both heavy and light methodologies may work for you – if you have the time to modify them to the needs of your organization and can make the effort to create the processes and procedures necessary to fill in the gaps. The On-Target 4D™ methodology is an attempt to address this issue and provide a step-by-step recipe for handling common business system oriented projects.

With nearly 20 years of experience in project management, I have been involved with hundreds of software development, commercial package integration, and network infrastructure projects. Some have been the work of a small handful of individuals working for a few weeks. Others have involved teams of over 200 individuals and have lasted a year or more. However, I have found that most of the software development

¹ For an introductory overview of Object theory and OOA/D practices, see the Introduction to Object-Oriented Analysis and Design with the Unified Modeling Language presentation on the Matincor, Inc. website at www.matincor.com

projects in the business world today involve teams of less than 2 dozen people and last between 3 to 6 months. (With the emphasis on duration continuing to push to the shorter end of that scale!) For that reason, the On-Target 4D™ methodology is targeted to those projects which:

- Have teams of less than 30 people
- Involve the development of custom business software
- Have a time frame of less than 6 months
- Are for corporate or general business environments (vs. government, military, or highly regulated industries)

The emphasis for the On-Target 4D™ methodology is to provide a pragmatic approach for a team with little or no formal training in software engineering. That is, a team which does not have extensive academic or practical knowledge of:

- Software development lifecycle management
- Use case development and analysis
- Class and object modeling at the problem domain level

The On-Target 4D™ methodology does use a subset of the Unified Modeling Language. A thorough discussion of the modeling notation is beyond the scope of this work. There are a number of excellent publications available which provide detailed information on this modeling notation. For the purposes of gaining a basic knowledge of the notation for use with this methodology, I would recommend “UML Distilled” by Martin Fowler, published by Addison-Wesley.

Guiding Principals

The On-Target 4D™ methodology (hereafter referred to as OT4D) is based on several basic principles:

Process Focus

The software development process takes precedence over detailed project plans. In fact, a well defined process in which the team is trained greatly reduces the dependency on monolithic project plans. The emphasis is on ensuring that team members understand the following:

- Processes which they own
- Processes in which they participate
- Acceptance criteria for artifacts which are inputs to their process
- Artifacts for which they are responsible
- Completion criteria for those artifacts

With a well defined process, the steps necessary to carry out any project are generally known at the beginning of the project. Custom, detailed plans are then developed for near-term activities – usually the next 2-4 weeks. No longer do you have project

managers working on the development of a project plan with hundreds, or even thousands, of interrelated tasks. Such plans are an exercise in frustration as they are out of date within days, or even hours, of completion. They also present a false sense of accuracy which may unintentionally mislead management and clients.

Team Knowledge

Effective software development requires a whole-team effort. The OT4D approach is focused on developing a common, team-wide understanding of the project objective, requirements, and design. Project artifacts such as scope documents, requirement lists, design mockups, etc. are by-products of the team effort. Consider these the project's equivalent of "meeting minutes". These artifacts are not goals in themselves; it's the team exercise in creating these artifacts that is the goal.

The various exercises that are part of the OT4D process are developed to build this team understanding. Most of the meetings and activities are attended by each of the team leads (representatives from each IT discipline). These meetings are extremely focused in scope, duration, and objective to ensure as effective use of the team's time as possible. This is intended to limit the "over the cubicle" workflow that unfortunately occurs with so many projects. Activities carried on in isolation, documented, and then "passed on" in the form of artifacts are a less effective way of providing the necessary team information required for rapid development.

The antithesis to the OT4D focus on team knowledge/experience is the "expert" who is too experienced and busy to spend time sharing his or her ideas and findings with lesser members of the team. Such a person may see themselves as the "Rambo" of the project. They keep knowledge to themselves – either as a form of power or because they are just not good communicators. This is especially dangerous when that person is in a lead role. Rambos don't win wars; effective teams do.

Clear, Simple Approach

Utilize the latest techniques that have proved themselves – but don't discard the tried and true just because they have been around a while. For example, the OT4D model is based upon an overlapping, incremental waterfall lifecycle model. Waterfall models have gotten a bad rap recently so we've been overwhelmed with development models that look like spirals, corkscrews, and water fountains. These all have important aspects which the creator was trying to communicate. However, we are dealing with extended project teams made up of people from multiple disciplines within IT not to mention the team members from the business side of the house (the "users"). Try to stick with tools and techniques that are easily understood. The OT4D methodology defines 4 key phases of a project; an approach which should be easy to communicate:

- Discover - What is it we want to create?
- Design - How are we going to build it?
- Develop - Build it.
- Deploy - Deliver it.

The overlapping waterfall lifecycle model of the OT4D methodology communicates the non-linear approach which we now use in software development. Aspects of analysis will continue while we are designing the system and development can occur while we are still designing. But the basic approach is understood by everyone without having to introduce new terminology and new concepts. We reduce risk thru shortened product release cycles – not by introducing new and confusing models which half the team doesn't understand.

These basic principles; a defined process, shared team knowledge, and easily understood concepts and tools, will help us to develop effective software products. They don't guarantee success but without this approach, your likelihood of success is significantly diminished.

The Process

A software development methodology has 2 components; processes and tools. The OT4D methodology is broken down for each sub-discipline within the standard IT department including:

- Project Management
- Analysis
- Development
- Quality Assurance
- Configuration Management
- Infrastructure

The detailed processes and tools used by each of these disciplines are described in separate sections of this document. An overview of the process is presented in two different charts:

- Activity/Artifact Matrix
- Process Flow Chart

At the highest, most simplified level, the process would unfold this way:

Discover	<ol style="list-style-type: none"> 1. Conduct 1 or 2 brief meetings with the project sponsor or domain expert. Identify high level functions and project/system boundaries. Produce a scope or vision document. 2. Build the project team. 3. Initiate a requirements gathering effort utilizing interviews, workshops, role playing exercises, questionnaires, market surveys, etc. Capture results in the Requirements document. 4. Conduct <i>initial</i> design meetings with domain experts and team leads. Capture results in Conceptual Design document including Use Cases.
----------	---

Design	<ol style="list-style-type: none"> 1. Conduct <i>detailed</i> design meetings with domain experts and team leads. Capture results in Conceptual Design document including Use Cases. 2. Conduct logical and physical design exercises to remove ambiguity and begin translation from text-based scenarios to design notation models. 3. Prepare test cases and scripts 4. Determine development, testing, and production environment requirements.
Develop	<ol style="list-style-type: none"> 1. Prepare work packages, including final physical designs, and assign application and coding jobs 2. Write the code 3. Prepare database conversion routines 4. Prepare installation and deployment scripts 5. Conduct unit and integration tests in development environment 6. Conduct system tests in QA environment and stabilize system
Deploy	<ol style="list-style-type: none"> 1. Conduct certification testing 2. Finalize release notes and other documentation 3. Conduct user acceptance and beta tests 4. Deploy 5. Conduct post deployment evaluation

The difficulty in representing an iterative process on paper is that you have a multi-dimensional process displayed on a 2 dimensional medium. An even if you could adequately display the iterative approach, it would tend to confuse rather than elucidate. Therefore, the table above and the referenced charts are very linear in their appearance. They describe a single pass through the process.

Iterative development with incremental releases is achieved thru the repetition of the lifecycle. This is an important point as the process is focused on product delivery, even if only in the form of an internal release. Object oriented development is an “additive” process. Like building a pyramid, each layer is added one at a time. If one layer is not stable, the subsequent layers will fail.

Plan on at least 3 iterations of the development lifecycle for all but the simplest of projects. The emphasis for each iteration is as follows:

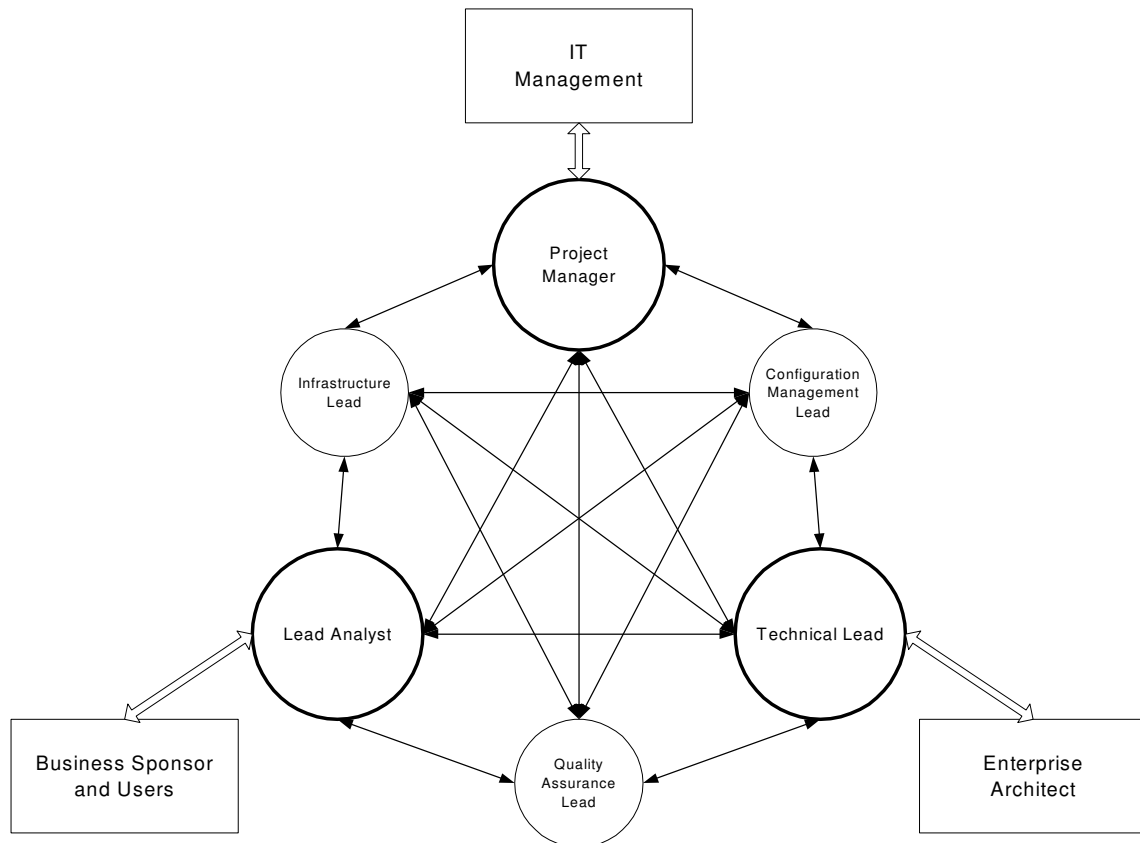
1. Focus on architecturally significant items such as core functionality, and non-functional and qualitative requirements. Address the high-risk items first.
2. Focus on correcting architectural problems and issues of robustness identified in the previous iteration. Include additional functionality.
3. Focus on secondary functionality and system stabilization.

The goal of each iteration is to create a demonstrable application. By the third iteration, a product should be available for user testing outside the development environment. Depending upon the project and progress, this may be an alpha, beta, or User Acceptance Test release. More complex projects will require additional iterations. However, beware; it is possible to enter an endless design and build loop that never delivers as we are always adding “just a little more” to prepare the perfect product. A “perfect” product risks infinite cost and infinite duration.

Team Structure

The team structure may be one of the most flexible parts of any project. A team is made up of many individuals with different skills, experiences, titles, responsibilities, etc. There is no single team structure that is correct for every situation. However, each software development project effort does demand certain activities which tend to be within the purview of certain roles. In the following sections of this work I will present the key roles and responsibilities of common team members. Depending on the size of the team and the experience of the individuals, these roles may be combined or split.

There are many team models in use today. Here is one take on a team model that has gained popularity recently:



This model communicates several important concepts. The first is that this is a matrix management structure. In this team structure, the Project Manager (PM) is not necessarily the administrative manager of the staff personnel. The PM runs the project, not the staff members. As a project manager, I have been responsible administratively for the staff as well as for the project, and at other times I have been responsible only for the management of the project. There are pros and cons to this situation but I have seen both structures work. It depends more on the maturity of the project team personnel than on anything else.

The model also points out the reporting relationship with extended project personnel.

Project Manager (PM)

In this model, the PM role is internal to the IT group and is not the primary communications conduit with the business. The PM reports to internal IT management or, sometimes, to an executive steering committee. The PM is responsible for letting senior IT management know the progress of the project and what issues have arisen which may impact the delivery of the final product. The PM also communicates with other PM's in the group to coordinate shared resources, release dates, and requirements

or design changes which may impact other projects. The PM is the backup to the Lead Analyst.

Lead Analyst (LA)

The Lead Analyst interfaces with the business sponsor and users. The LA represents the Users’ interests to the project team. The LA is also the team’s representative to the Users. They are responsible for managing user expectations – an activity which is every bit as important as requirements management or change management. The LA is the backup to the Project Manager.

Technical Lead (TA)

The Technical Lead, also sometimes referred to as the Development Lead, is responsible for the technical design and implementation of the solution. The TL reviews their design solution with the IT organization’s architecture standards group (Enterprise Architect) to ensure that a design solution is in compliance with standards. The TA is the backup to the Configuration Manager.

These 3 positions are considered the key positions in the project team. The remaining roles also participate in every phase but may have limited attendance to meetings and discussions depending upon the project phase.

The following table presents a brief overview of communication responsibilities for each team role.

Role	Internal Communication	External Communication
Project Manager	As the project overseer coordinates communications between the various team members.	Acting as the “eyes and ears” of senior IT management, reports on project status to the Executive Steering Committee.
Lead Analyst	Representing the client’s interests. They provide the understanding of the requirements and the vision of the solution.	Acting as the key liaison between the project team and the business, they represent the project to the business sponsor and users. The key responsibility is expectation management.
Technical Lead	Defines and drives the physical creation of a product. Lets the other team members know how the system will be designed and built.	Will review system design with the IT architectural standards group to ensure that enterprise standards are followed. If the project ventures into new territory for which no technical standards exist, they will work

Role	Internal Communication	External Communication
		with the standards group for a new standard or an approved variance.
Quality Assurance Lead	Reports to the team on the “health” of the product through Testing Results reports.	Works with the users during the User Acceptance Test. Obtains feedback on user test results.
Infrastructure Lead	Reports to team on status of infrastructure changes.	Works with telecommunications and other external communications service providers.
Configuration Management	Reports on status of baselined artifacts during build and promotion process.	Coordinates product promotion with Production Operations during the implementation phase.

Turn to the additional sections for a more thorough discussion of each role’s communications responsibilities.

II. Project Management

The Project Manager (PM) role is the most important role on the project team. The PM is the heart and soul of the project team and without them, the team can not hope to be successful.

The PM owns the following processes:

- Development and maintenance of the project plan and schedule
- Estimating and tracking project costs
- Providing project status to the project team and management
- Tracking project issues and resolutions
- Identifying and managing risk
- Negotiating and acquiring resources
- Coordinating resources and schedules with other project managers
- Managing outsourced project deliverables

The PM participates in most other project processes but is a significant contributor in:

- Project reviews with the business
- Tracking baseline project artifacts

Key artifacts for which the project manager is responsible include:

- Project Plan
- Project Budget and Cost Reports
- Contact List
- Issues Log
- Risk Matrix
- Project Status Report
- Project Notebook (which contains all the above)

Project Planning

The OT4D methodology defines the processes necessary for the creation of the various artifacts to be produced. The project plan under the OT4D process is not task oriented. It is deliverable oriented. The tasks necessary to achieve the deliverables are defined in the OT4D process and need not be repeated in the project plan. As the project's leads become more experienced with the execution of their respective processes and the ability

“The plan is nothing. Planning is everything.”

-Dwight Eisenhower

of their teams², they become more proficient at estimating the time necessary to achieve their deliverables. The PM negotiates with the team leads to coordinate the completion and delivery of their artifacts. The plan reflects artifact deliverable dates, not the activities necessary to deliver the artifacts.

This project planning approach contrasts markedly with the work breakdown structure (WBS) that is done in traditional project planning. Where no defined process exists, it must be created anew and detailed for each project; hence the detailed task, sub-task, sub-sub-task of the WBS model. In systems where deliverables require many weeks or months of effort, a WBS is required to keep team members focused on weekly activities and to allow the PM to measure progress. With an iterative approach to system development, a single pass of the complete OT4D lifecycle may be executed in a matter of weeks – 3 months at most. The individual process artifacts are created in a matter of days or weeks. In this way, the detailed WBS is not necessary and is, in fact, a hindrance. The focus is on delivering a complete system build in fairly short order. If the complete system can not be produced in these time frames, then a sub-set is built followed by additional iterations of the lifecycle process. In this manner, a large system will be built in a modular manner. This multi-cycle process is discussed further in the Analysis and Development Team sections of this document.

When a project is initiated and the analysis section of the Scope Document is drafted, the PM will convene a project planning meeting with a representative from each of the teams. The goal of the meeting is to discuss whether there are any factors evident which would require a change to the standard project process and to determine the general scale of the project. At his early stage, little is known about the detailed functional requirements of the system although system boundaries and scope should be fairly well defined. The scaling of the project may be to estimate whether the overall duration will be weeks, months, or quarters. (Should the project appear to be a scale of years, it is recommended that it be sub-divided for a more modular approach.) An initial schedule is created reflecting the estimated number of cycle iterations, the major stages of the OT4D process scaled to fit the cycles, and the size of the requested project team. This preliminary schedule is used in the creation of the initial cost estimate. The schedule is then added to the Scope Document which is then finalized. The Scope Document is discussed further in the Analysis Team section of this document.

Depending upon the level of Configuration Management and Change Control in your organization, the schedule may be baselined at this point. It is not generally recommended as it will almost certainly change significantly in the next few days or weeks as the Discovery Stage progresses and we have not initiated the Requirements Management process at this stage in the lifecycle. For a further discussion on these issues, please see the Configuration Management Team section of this document.

² The term “team” refers to the various IT departments, or disciplines, such as Analysis, QA, Development, etc. The term “project team” refers to the full team of staff assigned to a project.

The Project Manager will maintain the schedule and review an update with management or team members at the weekly or bi-weekly project review meeting. The focus on these reviews is the next schedule “window” of 1 or 2 weeks. As each iteration of the project life cycle is reached, a formal review of the full project plan will be conducted between the project team, IT management, and the business sponsor. At this time the project plan will be updated and formally recast with new time and cost estimates. The project plan will have a new baseline set following each of these reviews.

🔗 The Project Plan is a CM artifact and eligible for baseline and Change Management control.

Estimating and Tracking Costs

Estimating and tracking costs is probably the single most difficult and time consuming activity in which most Project Managers are involved. The estimating process begins with the Scope document in the initial Discovery phase and continues thru to at least the beginning of the last development iteration. Cost tracking begins immediately and continues through the entire project and completes with the final project review. Project costs usually include internal staff time and cost, contractor cost, and capital asset purchases.

There are two basic ways that project costs have traditionally been estimated. The first is by creating a monolithic project plan with a detailed work breakdown structure (WBS). You add estimated time and resources to each task and hit the “sum” key. This method depended upon a vast number of assumptions adding up to something reasonably close to the final cost – which it almost never did except for the smallest of projects. More disciplined approaches included methods such as function point analysis or lines-of-code estimates. While much more accurate, these approaches required detailed design specifications before an estimate could be made. In software development, by the time you’ve got detailed design specifications; your project is usually more than 2/3 complete.

In all but the simplest of projects, the software development process follows a Discovery, Design, Develop, and Deploy process repeated thru several iterations. At each iteration, our knowledge and understanding of the product requirements are refined which contributes to a better understanding of the development effort. The estimating process must follow the same model. An initial estimate will be made during the project scope effort based upon the team’s experience with similar projects. For smaller projects (measured in weeks), this estimate may be for the overall project. For larger projects, this estimate may only be for the initial Discovery stage with a more refined estimate to be delivered at the end of that stage.

In general, the estimating process will follow these steps:

- After the Analyst has drafted the Scope document, the PM calls a Scope Review meeting. A lead from each of the IT teams (Analysis, Development, QA, etc.) attends and reviews the Scope document. This group estimates the overall

duration of the project on a scale of weeks, months, or quarters. If the estimate is for longer than 6 months, consider breaking the effort into multiple projects.

- The group next estimates the size of the team necessary for the project. Realistically, the team choice is usually limited due to staff quantity, availability, and skill capability. Another factor to be considered is the amount of time the staff members can spend on the project as most IT personnel are assigned to multiple projects plus production support. (See the OT4D staff planning model for assistance with this process.)
- The estimating group makes an assumption about capital purchases; e.g. software licenses, hardware.
- An initial project iteration duration (Discovery, Design, Develop, Deploy) is defined. This process is commonly known as “time boxing” – a limit is placed on the amount of time that will be spent on a single iteration. Because the effort is time boxed, the cost of this initial iteration is very closely known and can be controlled. Further discussions regarding the iterative development process and time boxing are contained in other sections of this document.
- At the completion of the initial project iteration, a project estimating meeting with the team leads is held. The project cost estimate is recast and discussed with management. This cost estimate refinement process is repeated at the end of each iteration of the lifecycle.

This estimating process does not make the initial estimate any more accurate than the traditional WBS method. However, it requires much less up front effort and recognizes the inaccuracy of attempting a complete estimate so early in the lifecycle. In addition, the use of time boxing helps to control costs. This logic, however, may not convince your CFO to approve funding for the project. The only way to get a reasonable, lower risk estimate for a software development effort is to limit the size of the effort. For this reason, projects should be limited to only a few months of effort at most. Larger projects should be sub-divided into smaller ones. A 10% variance on a \$250,000 project is easier to deal with than a 10% variance on a multi-million dollar project.

Once the project is initiated and a budget approved, the Project Manager is responsible for reporting progress against budget. The project costs should be tracked and reported on a weekly basis to ensure that the burn rate will not exceed the targeted estimate. In most organizations, cost tracking is based on time tracking. You may wish to track time against artifacts as well. This historical data will assist the team leads in future estimating activities.

Status Reporting

The Project Manager is expected to provide formal project status updates to senior IT management on a weekly or bi-weekly basis as appropriate for each phase of the project's

life cycle. This is a summary meeting and may only include project managers and senior management. Such a summary meeting will follow the format dictated by management.

This section, however, is focused on a status gathering meeting which may or may not be attended by senior IT management. In this meeting, which includes the project leads, more detail will be discussed. This is not a meeting for the project manager to report to the team. Rather, this is a formal opportunity for the project leads to report to each other. The information gathered here will be used by the project manager in their summary report to IT management and the business project sponsor.

The focus of the meeting is to report the overall “health” of a project, the accomplishments for the prior period (one or two weeks), the plans for the next reporting period, and any critical issues which may be facing the project. Depending upon the criticality and progress being made, 2 project status meetings may be held in a week. The first meeting of the week is usually more in-depth. The second is briefer and only meant to report issues that may prevent each team from meeting its period goals. During the very later stages of a project, daily meetings may be necessary. Such meetings need to be tightly focused and controlled to limit their impact on the development effort.

Care should be taken to ensure that the status meeting does not devolve into an analysis or design meeting. The project manager must control the agenda and ensure that the status topics are covered. If further, more detailed discussion is desired, determine whether everyone needs to attend or whether a subset of the group will do.

Project status review meetings should include a representative from the user group for which the product is being developed. This key opportunity for 2-way communication keeps everyone up to date. It also provides a “training ground” for the user to gain a better understanding of IT’s development methodology and practices. However, there are times when issues which should remain internal to the IT group need to be discussed. This includes personnel and management issues. A separate meeting may have to be called for these discussions. (Like democracy and sausages, the details of project management are often upsetting to the non-practitioner!)

Project Issues

During the course of any project, issues will arise which may significantly impact a project from the perspective of:

- Schedule
- Cost
- Functionality

“*Significantly*” is usually described as something which will be “felt” by the end user of the system. A change in schedule which will impact a delivery date may impact the business’ ability to meet a market campaign, customer expectations, opportunity for a time-sensitive competitive advantage, etc. A change in cost beyond the project’s pre-approved variance allowance should be brought to management’s attention immediately.

And a change in the promised functionality for a product, or a specific release of a product, needs to be negotiated or coordinated with the business.

As the issues which may impact these factors arise, they should be identified, documented, and tracked. An owner for the issue's resolution should be identified and that person reports on the status to the project team as appropriate but not less than once per week. As each issue is resolved, it is the responsibility of the project manager to document the resolution and note the closure of the issue. The issues are tracked in the Issues Log section of the Project Notebook. For projects with a limited number of issues, the log may be appended to the weekly Project Status agenda.

Risk Management

The purpose of Risk Management is to identify those factors which may have a significant impact on the success or failure of a project. Whereas the issues as discussed in the previous section arise during the day to day activities on the project and are addressed in a reactive mode, Risk Management is a proactive approach. This exercise is often undertaken to address the assumptions which are naturally made when first planning a project. The factors most frequently considered include:

- **Technical** This includes technology related factors which may have unknowns that could impact the team's ability to deliver as promised. Samples include:
 - The use of a new programming language
 - The introduction of a new device (e.g. web-enabled phones, Persona Data Assistants (PDA's), etc.)

- **Resource** This includes personnel as well as physical resources. Samples include:
 - New staff who are not familiar with the software development process
 - Delays in computer hardware availability

- **Business** This includes changes in the business environment which may impact the project. Samples include:
 - Introduction of a new service by a competitor that requires significant change to a product under construction
 - Limited access to senior business management necessary for making critical product scope decisions

The Project Manager will work with the project team for the initial identification of risks which may impact the project. The risks are catalogued in the Risk Matrix which is located in the Project Notebook. The Risk Matrix is initially presented in the Scope Document and is formally reviewed at the end of each lifecycle iteration.

As each item is identified it is evaluated on several scales. All weighting is identified as Low, Medium, or High. The evaluated scales include:

- Likelihood This reflects the probability of the event occurring.
- Seriousness This reflects the possible impact on the project or organization.
- Difficulty of Detection This reflects how difficult it will be to identify when the risk is eminent or has occurred

The remainder of the Risk Matrix includes an area for documenting

- The impact should the event occur
- Preventative steps which may be taken in advance
- Contingency activities should the event occur or become eminent
- Identification of the trigger which could initiate the event
- The owner of the risk who is responsible for monitoring the trigger and initiating the contingency plan

Not all risks will have all this information pre-defined. The following table is a *recommended* approach:

Likelihood	Seriousness	Difficulty of Detection	Recommendation
High	High-Medium	Any	Document thoroughly
High	Low	Any	Emphasis on contingency plans and trigger monitoring
Medium	High-Medium	High-Medium	Emphasis on prevention and trigger monitoring
Medium	High-Medium	Low	Emphasis on prevention and contingency plans
Low	High-Medium	Any	Emphasis on trigger monitoring and contingency plans
Low	Low	Any	Emphasis on trigger monitoring

Negotiating Internal Resources

The Project Manager will work with the project team to identify resource requirements both in personnel and equipment. The Project Manager will see that necessary equipment is acquired and negotiate with the various heads of departments for human resources. During the life of the project the requirements for physical and personnel resources may change. It is the responsibility of the Project Manager to track these changing requirements and negotiate with management for the appropriate resources.

Tracking Baselined Artifacts

Artifacts are any standard or critical creation of the project team other than the system code itself. Examples include Requirements, Design Diagrams, Quality Assurance Test Scripts, etc. Items which are identified as coming under the control of the Change Control process are “baselined”. This usually occurs when a document is formally published. The reason for baselining a document is that such documents are milestones in a project’s progress through the software development life cycle. Such documents are key resources for activities following their creation. Should such a document require changes, it will very likely impact activities and deliverables which naturally follow it in the life cycle. These changes must be controlled and communicated to the appropriate team, extended team, or other stakeholders. The Change Control Process allows this to occur. Baselining artifacts also allows for a reconstruction of a product or project as it existed at key critical points in the past. This may be necessary should a return to a previous point in a project’s life cycle be required.

When an artifact is published or completed, the Project Manager will “baseline” the artifact. This will usually include having the Configuration Management Lead tag and store the object in a version control management system such as Microsoft’s® Visual SourceSafe. Should the project team determine that a change to the document be necessary, the Project Manager will follow the formal Change Control Process.

Another reason for baselining artifacts is that it is proof that a milestone has been reached.

See the Configuration Management section of this document for further discussion on this topic.

Managing Outsourced Activities

Should outsourced resources be utilized, the Project Manager is responsible to assigning activities to, and tracking deliverables from, these outsourced resources. An outsourced resource is any person or service provider that is not under the direct control of the IT management structure. The Project Manager is responsible for escalating non-performance issues to management.

Project Notebook

The PM is responsible for maintaining the Project Notebook. This was traditionally a 3-ring binder in which the PM kept project plans, a team roster, status reports, baseline

artifacts, etc. With the availability of shared network drives and easy-to-create web pages, this should now be a hyperlinked document available to all members of the project team.

III. Analysis

The Lead Analyst (LA) role is the most important role on the project team. The LA is the heart and soul of the project team and without them, the team can not hope to be successful.

The LA owns the following processes:

- Capturing and analyzing user requirements
- Designing conceptual solutions to address user's requirements
- Representing client interests to the project team
- Managing client expectations
- Product management

The Analyst also participates in processes that are owned by other members of the team. This may include working with:

- Designer to construct a system model
- Quality Assurance to conduct system tests
- Project Manager to prepare and update project plan as well as baseline Analysis artifacts
- Infrastructure to define development, test, and production environments
- Users during product deployment

Key artifacts for which the Analyst is responsible include:

- Scope
- Requirements
- Conceptual Design
- Release Notes
- User Guide and Online Help Screens (may be responsibility of technical writer)

Role and Responsibilities

First and foremost, the Analyst is expected to work with the client to identify and capture their requirements. These may be associated with the replacement of a current system (whether manual or automated), a new system requirement, or the enhancement of a current system. In some cases, the Analyst will work with business management to gather the requirements on a generic system that will eventually be modified for different divisions of the company or for customers of the company. In such cases, the Analyst will capture the core requirements initially and then, as each division or customer is addressed, capture the unique requirements associated with each. Usually, the application effort undertaken for each division or customer is addressed as a separate project with its own plans and deliverables.

System requirements are reflected in 2 key artifacts for which the Lead Analyst has primary responsibility.

Scope Document

The project's Scope Document is the initial product of the initial Discovery stage of a project. When a project is initiated, there should be a meeting or conference to define the project's goal. This meeting should include the LA, the business project sponsor, and the business user (domain expert) who will be the key participant on the project team. Other participants may include the Project Manager, a senior IT manager (especially for mission critical or high cost projects), and the Technical Lead (especially if the use of new technology is anticipated). The Scope Document is used to capture and reflect the points discussed in this meeting and will reflect an "executive level" understanding of the project. Usually this initial meeting has a duration of only 1 or 2 hours, a half day at most. If the meeting lasts longer, it is probably going beyond a project scope discussion and evolving into a requirements gathering meeting. There is no problem with this; it just needs to be recognized that the information gathered may not all be reflected in the Scope Document.

The goal of the Scope Document is to ensure that there is a common and consistent understanding of the "vision" of the project by all stakeholders prior to the significant undertaking of the requirements gathering activity. The Scope Document contains the following information:

- Basic, high-level system functionality
- List of user roles (Actors)
- List of anticipated system interfaces
- System availability
- Key assumptions including cost or time restrictions
- Critical success factors
- Conceptual domain model (Use Case Diagram)
- Is / is not matrix

The objective for gathering this information is to establish project boundaries. As the project progresses through the Discovery phase, more detail for the high level system functionality will be discovered and explained but the original function list should change little if at all.

The Scope Document will also contain a section which is to be produced by the Project Manager. This section will include an executive-level project approach statement. The project approach statement will drive the project plan which will be presented at a later date. The section will also contain an initial risk assessment and cost estimate.

The Scope Document is intended to be a static document. It is created at the beginning of the initial project iteration only and will not usually be revised in future iterations. Minor modifications to a project's scope may be addressed in the Requirements Document.

Should significant changes be required at the Scope level, they are probably so fundamental as to change the entire approach of the project. In this case, the project may need to be re-initialized. A simplified analogy might be the construction of a building. At the Scope level of the project, you are defining whether you are architecting an office building, a beachside hotel, or a single-family log cabin. If you change the type of building, that change will be very fundamental to your project.

For a sample template with outline and section explanations, please see the Scope Document Template.

🔒 The Scope Document is a CM artifact and eligible for baseline and Change Management control.

Requirements

The creation of the first iteration of this document is a key activity conducted during the Discovery phase of the software development life cycle. However, unlike the Scope Document, this is a dynamic product that will evolve during the life of the project. A key responsibility of the extended project team³ is to determine when the Requirements are sufficiently known so that the next phase in the life cycle may begin. This may also involve defining project iterations so that one sub-set of requirements may be moved to Design and targeted for an initial release and that incomplete requirements may be targeted for a later project iteration. It is not expected that the Requirements Document be completed and “locked down” prior to the next phase of the life cycle beginning. It *is* expected that, prior to beginning the Design effort, we have a sufficient understanding of the requirements so that we can:

- Begin a system-level architecture
- Define infrastructure needs
- Map out key functional areas that may be aligned with iterative development efforts or phased releases of the product

The Requirements Document will contain both functional and non-functional requirements. There will also be sections of the document authored by team members other than analysts. For a sample template with outline and section explanations, please see the Requirements Document Template.

🔒 The Requirements Document is a CM artifact and eligible for baseline and Change Management control.

Several standard analysis processes will be used during the capture and analysis of requirements. These may include individual and group meetings to develop a requirements “checklist”, questionnaires for a wider audience, role-playing exercises to

³ The extended project team includes business representatives.

capture workflows, research to discover services offered or planned by competitors, evaluation of the current system (including interviews w/ Customer Support), etc.

Conceptual Design

The result of the analysis of the requirements will be a proposed system or solution. The vehicle for communicating this proposed solution is the Conceptual Design Document. For the very simplest of projects, this may be nothing more than User Interface Specifications with screen flows. It may also include workflows in the form of flowcharts or scenario diagrams, business rule algorithms, report layouts, functional (system) schematics, user-level entity relationship diagrams (user recognizable entities such as customer or account), etc. For more complex systems, there will be full Use Cases with both success and failure scenarios defined. The level of documentation will be dependent upon the type and complexity of system and agreed upon by the project team.

See the sample Conceptual Design Document for an example with Use Cases.

There are several key sources of information for the Conceptual Design document:

- Scope Document
- Requirements Document
- Joint Application Design (JAD) sessions
- Individual interviews
- Market and industry research
- Evaluation of current system
- Resource materials from similar applications

Evaluating the current system is part of the standard Discovery process that an Analyst is expected to undertake. The current system may be either manual or automated. And evaluating resource materials may include looking at competitors' products or similar commercial-off-the-shelf (COTS) products. JAD sessions and Use Cases, however, may be new to the team or individual staff members.

Joint Application Design

A Joint Application Design (JAD) meeting is a forum for getting users and IT staff together in a venue where open and frank discussions are possible. The term is used broadly here and refers to any large group meeting between the users and the IT team convened for the purpose of information gathering and evaluation. The objective of a JAD session is to provide another method for the capture and evaluation of system requirements. It does not replace individual interviews, current system evaluations, or other research. The JAD does, however, draw personnel together where they may share and openly discuss ideas, concerns, questions, and suggestions regardless of their rank or role within the organization.

JAD sessions should include several members from the user community. You will need more than one to help reduce the risk that your system ends up being developed for the particular way a single user performs their job. This opportunity for different users at various levels, including management, to get together and discuss the issues is invaluable. Often these sessions will point out issues with current business processes and differences in perception between management and staff.

A JAD meeting should be limited to 8-12 personnel with IT not outnumbering users. If the users feel an overwhelming presence of IT personnel, they may be uncomfortable speaking openly. The objective is to make them as comfortable as possible to create an environment for open communication. This meeting is facilitated by the Lead Analyst or other IT staff comfortable in this role. IT team members should include the analyst(s) assigned to the functions under discussion, the Technical Lead, and the Project Manager. Optional attendees include the Quality Assurance Lead and the Infrastructure Lead although their presence may be beneficial during the highest level discussions. Configuration Management does not usually participate in these meetings unless the discussions include deployment planning.

The JAD is not a “brain storming session” although that activity may be a component of a session. The JAD is structured with an agenda, a goal, and a scope of discussion. There may be several JAD sessions during the analysis of a single Use Case or system function requirement. The JAD is also not a “silver bullet” that will address all the risks and problems naturally inherent in human communications. It is another tool which may be used and, like any tool, it needs to be used at the correct time under the correct circumstances. The decision to utilize the tool is made by the project team.

A JAD session is particularly useful for defining the user interface and the interaction between a user and the system.

There is a great deal of information available regarding conducting and utilizing JAD sessions. A little Internet research can provide some basic material and books are available with more detail.

Use Cases

Use Cases are a way of capturing functional requirements and will frequently be the “output” of a JAD session. Many functions are best described in narrative or sequential activity form. Use Cases provide a format for documenting these “stories”. UC’s may be short; an informal, brief paragraph describing a user action and their experience using the system. UC’s may also be more lengthy and captured in a specific format. UC’s may be used anytime that a sequence of user/system interactions need to be captured. They may be used to capture

current processes but most commonly they are used to reflect how a system under design will work.

For more information, see the OT4D presentation “Building Use Cases” and the Conceptual Design Document template which provide more detail on how the Use Case format is used in this process. A thorough discussion of Use Cases is beyond the scope of this work. There is a great deal of published information available discussing the creation and use of Use Cases and more is being added daily. A little research should provide you with all the detail you would like on the topic.

Special Note: I do disagree with one element of the Use Case approach which is generally accepted. That is the inclusion of a discussion of user interfaces. Many works that I have read regarding Use Cases says that there should be little or no consideration of user interfaces or user interface design at this phase in the project. Unless you are designing a system with a completely new technology, you probably already have an idea of how the user interface is going to work. I have found that by drawing mockups on a whiteboard or flip chart, it makes it easier for people to communicate their ideas. Realize that if you have a dozen people in the meeting, unless you draw a mockup on the wall, you will have 12 different opinions of what the interface will look like. That is a user expectation that must be managed. This is not to say that a detailed user interface design should be created, only a mockup or outline. Let the usability engineers do the formal design.

The meetings to gather requirements and capture Use Cases are owned by the Analyst team. They are usually facilitated by the Lead Analyst but pick a good facilitator regardless of their formal role on the team. These meetings will be attended by subject area experts (users), the analysts, and the Technical Lead. Optionally, they should also be attended by the Project Manager, Quality Assurance Lead, and Infrastructure Lead. These secondary attendees are particularly important during the initial, high-level Use Case discussions. However, as sub-Use Cases are discussed, their attendance may not be as necessary. The Configuration Management Lead is not required for these meetings.

In my experience these meetings should be at least 2 hours long and not exceed 4 hours. It often takes half an hour for people to open up and start contributing and it is not until the second hour that things really get rolling. By 3-4 hours however, most people are completely drained and need a break. Provide snacks, sodas, and coffee; they help keep the energy level up and getting people to move around sets a more informal tone to the meeting. Make people comfortable to improve their creativity and concentration.

The Conceptual Design Document, including the Use Cases, is the single most critical non-code artifact of the OT4D methodology. More than any other artifact, this one impacts almost all the downstream activities and artifacts to be produced during the project life cycle. These include:

- Physical design including business logic and user interface design
- Quality assurance test cases and scripts
- Infrastructure server and network impact analysis
- Training and User Guides
- Business process re-engineering

It is usually during the conceptual design activity in the Discovery phase of a project that consideration will be given to a phased release of the product. This will include defining what features and functions will be included in each release and the relative timing of the releases. This will also provide the opportunity, if desired, to split the project team into those focused on the immediate release and those focused on a later release.

🔗 The Conceptual Design Document(s) is a CM artifact and eligible for baseline and Change Management control.

Release Notes

Software development is a difficult process and any significantly complex system will have defects even when released to the production environment. When a product becomes eligible for release, Release Notes will be required. This is true whether for an internal iteration, an alpha or beta release, or for a full production release. Release Notes detail known defects by function. If known, workarounds will also be included.

User Guide and Online Help

User guides and online help are frequently afterthoughts unless created for a commercial product. Commercial products will probably have professional writers who will prepare the User Guide. If the product is for internal use by the company, it should have at least online help. Often the only staff capable of producing this documentation are the analysts. This should be discussed during the initial project scope meeting as it can be a significant undertaking and impact project cost and duration.

Client Representation

A key responsibility of the Analyst is the representation of the user's interest in the project. The Analyst ensures that the system addresses the needs of the user. When the project team is discussing how to best design, develop, or implement the system, the Analyst is responsible for seeing that the interests of the user are being represented.

Expectation Management

The opposite side of the Client Representative "coin" is the representation of the IT team to the user. The Analyst will help manage the expectations of the user by keeping them informed of design decisions, functions addressed by each release, etc. This may sound like the project status reports supplied by the Project Manager, however there is a key difference. The Project Manager represents the "project" including schedules, costs, etc.

The Analyst represents the “product” including what requirements will be addressed by various versions, technical restrictions, etc.

Product Management

The Product Lead is the IT staff member responsible for the management of a complete product. The Product Lead is responsible for the IT internal vision and the long-term strategic planning for a product regardless of whether there is an active project or not. They have responsibility for the product throughout its life cycle.

The Product Lead is usually the senior-most Analyst on a project. This may not be a full-time assignment and there may be other, more junior level analysts who are assigned to the project full-time.

IV. Development

The development team is led by the Technical Lead (TL). The Technical Lead role is the most important role on the project team. The TL is the heart and soul of the project team and without them, the team can not hope to be successful.

The TL owns the following processes:

- The technical (physical) design of the application
- Preparation of development “work packages” for individual developers
- Actual system development including application coding and database creation
- Unit and Integration Testing
- Final versions of design documentation (“as built”)
- Creation of installation and/or deployment scripts

The TL also participates in processes that are owned by other members of the team. This may include working with:

- Project Manager to prepare and update project plan as well as baseline Development artifacts.
- Analysts to define Conceptual Design
- Quality Assurance to conduct system tests
- Infrastructure to define development, test, and production environments
- Enterprise Architect to obtain approval on the system design

Key artifacts for which the TL is responsible include:

- Design Documents & Models
- Work Package
- Application Code & Databases
- Data conversion routines
- Unit & Integration Test Cases
- Implementation Scripts

Role and Responsibilities

The Development group is responsible for the technical design and creation of the actual systems. A representative from Development will be assigned to a project early in the Discovery Phase. Their initial contribution is advisory in nature and the individual will assist the Analyst group in evaluating the requirements of the system. It is not until the Design Phase of a project that the Developer has primary responsibility for project artifacts.

There are usually several developers on a project. The senior-most developer on the project will fill the role of Technical Lead and will be the person who has overall

responsibility for the creation of the physical product. The Technical Lead's key activities include directing the physical design of the application and creating the Work Package. The Work Package contains the specific instructions for the development of code and is used to issue work orders to individual development team members. The Technical Lead, who may also develop code, will gather the units of work together for initial System Integration Testing.

In some IT organizations the role of designer, or system architect, and developer are separate roles. Additionally, on more complex projects, the design and development responsibility may be shared between those focused on the application code and those focused on the database and stored procedures. In such cases, a single individual should be assigned the Technical Lead role for communication routing purposes.

Discovery Phase

During the Discovery phase of a project, the TL's activities are more advisory in nature. The TL will participate in the initial team-led review of the Scope document which has been prepared by the Lead Analyst. They will assist in the team's discussion and decision regarding project approach and initial timing estimates.

When work begins on the Conceptual Design document, Use Case preparation will be initiated. The Use Case work sessions are lead by the Lead Analyst but the TL will want to participate for several reasons. The most obvious is that the TL can begin to develop an understanding of how the application will need to function for the users. They can gain a specific understanding, by key function, of required response time. Combine this understanding of the required functionality with the non-functional or qualitative requirements defined in the Requirements document, and the TL can begin to understand the factors which will impact their architectural decisions.

The TL can also help guide the team during the Use Case creation phase. The TL will often point out difficulties or challenges that they will face with various scenarios being discussed. While it is important to discuss a variety of options, even those that are "blue sky", the TL often can offer a dose of reality before the team goes too far down a path that is impractical. On the other hand, the TL can also gain insight into business requirements that might force an architecturally challenging approach.

By having the TL participate in the Discovery phase, the project can often gain a boost to its momentum. Architectural considerations and analysis can begin earlier than might otherwise be the case.

Design Phase

The first phase of a project for which the TL has primary responsibility for deliverables and artifacts is the Design phase. In the initial iteration of the project, the design effort should be focused upon those requirements which are architecturally significant or challenging. Requirements that commonly fall into this area include:

- Transaction oriented activities with an anticipated heavy volume or especially rapid response requirement.
- Requirements with complex logic.
- Transactions that will span multiple platforms – especially those spanning organizational domains which require distributed processing between heterogeneous computing platforms and security systems.
- Requirement that system be developed using new technologies.

An initial Requirements Document and Conceptual Design Document should have been created. As a team lead, the TL will have been involved in the creation and review of these documents. Remember, in the OT4D methodology these documents are by-products of the Discovery effort. The group exercises and discussions that led to their creation should have included the TL as well as the other team leads. The TL can use these artifacts to provide background information to the development team. This information will provide the input material for the Design phase and include functional and non-functional (qualitative) requirements as well as (business) domain-level Use Cases. As further iterations are undertaken, requirements will be more fully explored and refined and the design enhanced to provide a more robust technical architecture.

There are a wide variety of tools and techniques in use today to assist in the Design phase of a project. Subsets of these have been selected as the preferred tools and techniques for those types of projects targeted by the OT4D methodology. However, each project is different and has its own unique requirements. The Technical Lead is expected to select those approaches that will best meet the needs of the project.

Use Case Drill-Down

In the OT4D methodology, the transition from Discovery to Design occurs when the emphasis shifts from discussing *what* the system will do to *how* it will do it. This usually begins with the Use Case Drilldown sessions. The objective of the drill-down is to begin the “translation” of text-based requirements to a system model. The initial translation should help to identify and remove areas of ambiguity. This is a critical juncture in a project’s lifecycle and a mistake here will have a rippling effect through the rest of the project effort. Great care should be taken not to gloss over this exercise.

The Technical Lead is expected to lead this meeting. The objective of the meeting is to identify objects (and by extrapolation, classes) and the communication between those objects. In the early iterations of the project, the Use Cases will be at the domain (business) level and so, therefore, will the objects identified. While these high-level objects and classes will probably not be directly realized in the actual physical design of the system, their discovery and definition is nonetheless important. Objects at the business level will be recognized by the domain expert (the User). Changes to system requirements will tend to be in terms that can be expressed through these domain level objects. A system modeled on these high-level conceptualizations will be more easily modified in the future.

The session should begin with a review of the Use Case under consideration and any associated materials. The Use Case is then analyzed with a goal of identifying the essential domain level objects, inter-object communications, processes (methods), and data elements (attributes). During the initial analysis, this information may best be captured utilizing CRC cards. CRC stands for Class, Responsibility, and Collaboration and refers to sections of a paper card, commonly 3x5, upon which this information may be written⁴. As the design effort evolves through more detail and further iterations, a more thorough method for documenting the design discussions would include Unified Modeling Language (UML) Sequence Diagrams and Class Diagrams.

Examples of Sequence and Class diagrams may be found on multiple Internet sites including Rational University's UML Quick Reference page. Multiple Sequence Diagrams or diagram subsets may be required to effectively convey different Use Case scenarios including successful completions ("Happy Day" scenarios) and execution failures. However, it is not necessary to create a sequence diagram for every possible path of execution through the Use Case. The main path should be diagrammed in some detail. Alternate and Exception paths should be discussed and only complex or critical processes diagrammed.

The emphasis of the Use Case Drilldown exercise is the discovery and definition of system objects at the level of the Use Case under discussion. The UML Sequence and Class diagrams are a way to document these discoveries. It is not essential that the UML notation be followed explicitly unless you intend to generate code from the models. Again, the goal is team understanding, not diagramming every subtle nuance of the Use Case.

As the drill-down effort continues across the various Use Cases, sub-Use Cases may be identified. These will have to be defined and may have Sequence Diagrams created. As the Sequence Diagrams and Class Diagrams are created, patterns may begin to emerge. It is the responsibility of the designer, or system architect, to express these patterns in a design that will fulfill the needs of the domain level objects.

The design effort continues as the model evolves from a conceptual-level model to a physical-level model. When the initial drilldowns are conducted on domain-level use cases, the TL may request that the Lead Analyst and appropriate Analysis team members participate. If the domain expert is a member of the user community and has had exposure to the software development process, they may join as well. However, at some point the conceptual world of the domain level Use Cases is left behind as the design discussions turn to a more physical model. As the discussions continue, the non-developers on the project will have an ever lessening role in these meetings. However, the Lead Analyst will continue to be available to address questions or points of clarification that may arise.

⁴ There are several good papers on using CRC cards available on the Internet and I will not attempt to readdress this information here.

Design Documentation

As a result of information gathered during the Use Case Drill-Down meeting, the Technical Lead is expected to produce several artifacts. These generally include:

- Sequence Diagrams w/ inter-object messaging parameters
- Class Diagram with major methods, attributes, and collaborations identified
- Entity-Relationship diagram (if relational rather than OO database is to be used)
- Data Dictionary
- Process flowcharts for complex logic
- System architecture model overlaying an infrastructure topology map (optional and usually one per project rather than one per Use Case). This document is a combination of the Deployment and Component diagrams referred to in the UML.

The recommendation for process flowcharts is in contrast to the Activity and State Machine diagrams defined in the UML. The reason for this recommendation is that flowcharts are well understood by many people and have a much richer notation than the charts offered in the UML. This is a case where the “tried and true” may work better than the new. However, the same information can be conveyed using the UML if preferred.

Following the creation of the documents, they should be reviewed in a follow-on team meeting for further drill-down or approval. It is anticipated that complex Use Cases may require several iterations of the Drill-Down and Document cycle.

In some cases certain Design documents may be reviewed with the business - depending upon the complexity and quantity of documents generated and their relative level of detail. Many of these documents will be new to the business user and will require significant explanation. However, valuable feedback can often be obtained by these reviews.

Architecture Review

When the Technical Lead has prepared the architecture design models for the first iteration of the project, there should be a review meeting conducted with key team leads and any body responsible for design and development standards within the IT organization. The discussion should include a review of the non-functional and qualitative requirements. Using scenarios or examples for each listed requirement, the TL should be able to explain how the system design will meet these various criteria.

Development

The Technical Lead will initiate and supervise the development effort which occurs during the Development Phase of a project. They will be responsible for overseeing the work of the developers and ensuring that they comply with the organization’s coding

standards. (In the event that there is an Application Lead and a Database Lead they will each be responsible for their area.) As the individual developers complete their work, they are expected to perform Unit Testing to ensure that the code executes properly. As the units of work are completed, they will be brought together by the Technical Lead to perform Integration Testing. The Technical Lead is responsible for comparing the actual code artifact against design documents to ensure consistency. If there is a difference, one will be brought in line with the other to ensure that what is built is documented. This will ensure that accurate documentation is available for production support and maintenance efforts.

Work Package

The Development Phase of the project is initiated with the creation of a Work Package document. This presents the “work package” for the construction effort of the developers. This is the work order from which they will obtain their daily and weekly tasks. This may cover entire Use Case(s) or only a subset of the classes identified.

The Work Package provides the instructions for the developers detailing the work requested, the design models, and the names of the files or components affected (if pre-existing). The document is the responsibility of the Technical Lead. However, the Lead Analyst and Project Manager may contribute as well. The Technical Lead will assign the work package to a developer who is responsible for the creation of the code. A target date should be established for completion of the unit of work so that the Technical Lead can accurately update the Project Manager for work coordination and scheduling purposes.

Work Package Assignment Meeting

Generally, the Technical Lead will call a Work Package Assignment meeting. This may be an informal meeting between the Technical Lead and a developer or a more formal meeting with the full Development team. The purpose of the meeting is to review the Work Package document with the staff that will be performing the actual work. This is a chance for the Technical Lead to communicate his expectations and for the individual developers to ask questions. Following the meeting, the actual coding effort begins.

Coding & Unit Testing

This phase of the project includes the actual coding work. This includes application coding, database construction, stored procedure development, etc.

Each developer is responsible for his or her coding product. They are to ensure that the product of their work effort operates properly within the parameters defined by the designers. Developers are expected to perform Unit Testing of their work and to participate in System Integration testing performed by, or under the direction of, the Technical Lead.

When a developer finishes their work assignment on an individual component, they are expected to check it in to the project’s software repository (see the Configuration

Management section for further information). After the component is checked in, the developer will notify the Technical Lead that it is ready for System Integration Testing.

Code Walk-Thru

At this point, the Technical Lead may call for a Code Walk-Thru. This will occur on an “as-needed” basis at the discretion of the Technical Lead. There are several reasons that a Code Walk-Thru may be called:

- Ensure that the code complies with coding standards
- Ensure that the code is efficiently written and meets the design specifications
- Provide a training opportunity for junior members of the team by allowing them to see the construction of a more difficult component

System Integration Testing

When there are sufficient components to build a working (sub)system, the Technical Lead will conduct System Integration testing. The purpose is to ensure that the constructed or modified components work together properly. This testing is conducted in the development environment and is prior to QA testing. If the System Integration Test fails, the Technical Lead will ascertain the problem and either fix the problem or return the component to the developer.

Once the Technical Lead is satisfied with the results of the System Integration Test, he will notify the Project Manager and the QA Lead. If the functionality of the integrated components is sufficient for QA to initiate their tests, the release (sub)candidate is eligible for promotion to the QA staging (test) area. Prior to promotion, QA may request to test the product in the Development Environment.

Promotion to Quality Assurance Staging Environment

If the team decides to promote the system to QA, the System Promotion Procedure will be activated. Please see the Configuration Management section for further information.

Deployment Scripts

When the application is ready for promotion to the Quality Assurance test environment, the development team will create an installation package. This package will consist of automated installation scripts and installation instructions. These should be identical, or as close as possible, to those that will be used for the final production deployment. This will allow for testing of the deployment scripts as well as the application within the environment.

Before the application may be promoted to the QA test environment or to production, Configuration Management must be supplied with the complete package of code and associated project documentation. This full artifact collection is the responsibility of the Project Manager but the Technical Lead will be responsible for identifying the Development artifacts.

The product will now enter a “test and fix” cycle between Development and Quality Assurance. This cycle will continue until the product is deemed eligible for QA Certification Testing or promotion to the User Acceptance Test environment.

V. Quality Assurance

The Quality Assurance Lead (QA) role is the most important role on the project team. The QA Lead is the heart and soul of the project team and without them, the team can not hope to be successful.

The QA Lead owns the following processes:

- Defining the project's Test Strategy and Test Plan
- Preparing Test Cases
- Executing tests
- Reporting test results
- Defect tracking and reporting

The Quality Assurance Lead participates in processes that are owned by other members of the team. This may include working with:

- Analyst to obtain requirements for use in developing Test Scripts
- Project Manager to determine QA testing strategy and resource requirements
- Configuration Management to ensure testing of proper product version
- Configuration Management and Infrastructure to define test environments

Key artifacts for which the Quality Assurance Lead is responsible include:

- Test Plan
- Test Cases
- Test Results Report
- Product Certification Report

Role and Responsibilities

The Quality Assurance (QA) activity in the OT4D methodology is focused on measuring product stability and its compliance with documented requirements⁵. The approach for determining and measuring product quality is via tests conducted in a controlled environment to identify defects (programmatically errors) and variances from the various requirements and design documents. Errors and variances are tracked and reported. This tracking and reporting activity is made easier thru the use of defect-tracking databases. Such databases may be as simple as a spreadsheet or tracked in very sophisticated defect tracking tools which are available commercially.

⁵ A true Quality Assurance group will be responsible for much more than testing. This may include auditing the project team's compliance with the defined process, evaluating the various finalized project artifacts against published completion criteria, etc. The OT4D methodology for the QA activity focuses on the testing responsibilities only.

When a project is initiated, Quality Assurance resources will be assigned. Depending on the scope of the project, this may be a part-time assignment for an individual, or several QA staff members may be assigned. The senior-most QA analyst on the project is referred to as the QA Lead and is responsible for directing the activities of other QA staff assigned to the project.

Test Planning

The Quality Assurance participation on a project should be initiated in its earliest phases. The first activity in which QA participates is the team's review of the proposed project's Scope Document. This document, which should be reviewed by each functional leader on the team, presents the initial "vision" of a project. This document may contain a statement regarding the QA test strategy for the project. This is an extremely high-level indication of the type of testing that will be undertaken. This statement is not required and may only be used when a project is to have a different testing approach than prior such projects.

The first regular process undertaken by QA is the development of the Test Plan. The test plan effort is usually initiated during the Discovery Phase of a project. The goal of the test planning activity is to

- Identify the type of testing needed to address risk mitigation (examples include: function error discovery, load testing, system ergonomics, black-box vs. white-box, and volume testing)
- Determine QA staff, test tool, and test environment requirements
- Set team and management expectations as to the testing focus and thoroughness of product validation
- Initial identification of Test Cases that will need to be prepared
- Establish error tracking and reporting procedures

These items will be documented in the QA Test Plan which is included as part of, or an attachment to, the Conceptual Design document. The Test Plan is the primary document that introduces and outlines a given project from the QA perspective. The Test Plan provides criteria for measurement, deadlines for test artifacts, and describes procedures for documenting and resolving problems.

Full system tests are impractical in all but the most critical systems – such as life support systems. A complete and thorough test of all requirements is akin to asking for a 100% defect free software product with a guaranteed uptime of 100%. While an admirable goal, the cost approaches infinity. The purpose of the Test Plan is to identify which tests should be undertaken given the risks associated with failure of particular requirements. Select the tests which are practical given these risks and set expectations for the client, business sponsor, management, and the IT team.

The Test Plan, like the rest of the Conceptual Design document, is a living part of the project and will evolve during the project's lifecycles. This is especially true during the initial project iteration.

🔒 **The Test Plan Document is a CM artifact and eligible for baseline and Change Management control.**

Test Cases

Test Cases are designed to test each function and qualitative requirement of the system. Each Test Case should clearly define the expectations of the test, the procedures to execute the test, the expected results, and the actual results. A Test Case may be a document containing detailed testing steps or test scripts for use in an automated testing system. These test artifacts are based on the various requirements and design documents.

Those tests focused on system functionality will be aligned with the Use Cases. The Test Cases are prepared during the Development phase and in parallel with the actual application development effort. The preparation may require as much effort on the part of QA as the application development effort of the programmers. It is therefore critical that changes to requirements and design be communicated to the QA team as soon as possible. The process for this communication is described in the Configuration Management Processes documents.

Tests focused on qualitative, or non-functional, requirements should also be considered. Some tests may be more practical, and easier to execute, than others. There are a variety of testing tools available which can simulate load, volume, and other stress tests. However, testing for requirements such as portability and modifiability are more difficult to undertake. Portability testing may require multiple test environments or multiple re-configurations of a single physical test environment. Modifiability may be addressed through post-construction ("as build") architectural review analysis. On a practical basis, actual modifiability will not be realized until later system modification efforts.

🔒 **The Test Case documents are CM artifacts and eligible for baseline and Change Management control.**

Test Execution

The Quality Assurance Lead and their team are responsible for the execution of the tests that will measure the stability and requirements compliance of the system. There are several different types of tests which may be conducted. These differ in scope, thoroughness, and test environment. The Unit Tests and Integration Tests noted below are part of the Development process and therefore owned by the Technical Lead. However, they are included here to fully describe the testing cycle. The purpose of the multiple tests is to discover defects as early in the development cycle as possible. Tests include:

Unit testing

Unit tests consist of testing individual programs or components as they are written instead of testing after they have been integrated with other system components. The testing of smaller building blocks is done first and then these blocks of code (collections of classes and components) are combined and tested. The Developers usually perform unit testing, but there may be cases in which Quality Assurance is involved with Unit Testing. Unit Testing is conducted in the Development environment.

System Integration Testing

An Integration Test is a series of tests whose primary purpose is to exercise the system to uncover errors introduced by the inclusion of new code. This is an orderly progression of testing in which software and/or hardware elements are combined and tested until the entire system has been integrated. The purpose is to measure the correctness of each program unit's behavior once the code has been combined with other components. The Developers usually perform the initial Integration Testing under the direction of the Technical Lead. Quality Assurance may be involved in Integration Testing to pre-validate Test Cases or the system product prior to promotion to the controlled Quality Assurance environment. Integration Testing is conducted in the Development environment.

“Test & Fix” Validation Testing

Once a product is promoted to the QA test environment, it undergoes a series of formal tests utilizing the prepared Test Cases. These tests include both functional and non-functional requirement testing. The express purpose of the tests is to identify and report defects which are then immediately addressed by the Development team. This may require an iterative code, test, re-code, re-test cycle to evolve the code into a stable product. This cycle will continue until the project team feels that sufficient stability has been achieved to have a release candidate. The duration of these cycles will depend not only upon the coding effort but the development-to-QA promotion steps including system builds and the re-baseline of the testing environment (discussed further below).

Several testing phases may occur in the QA test environment. The first is to determine whether the system was properly installed. The Configuration Management Lead will conduct a system build and run installation routines in order to execute a controlled promotion of the product to the test environment. The initial test, sometimes referred to as a “Smoke Test”, is to determine whether the installation was successful and the system stable enough to execute a formal, controlled test. This initial test is informal and may only require a few minutes or a few hours depending upon the size of the test candidate.

With a successful execution of the Smoke Test, the QA Lead will initiate a full system test per the test plan. This test is formal, structured, and managed. The use of automated testing tools is extremely helpful during the effort. This testing effort will normally include regression testing. Regression testing is the selective retesting of a system or a component to verify that modifications have not caused unintended effects and that the system or components still work as specified. An automated testing tool is required to efficiently perform regression testing on large-scale projects.

Certification Testing

Once a product has been identified as a production release candidate, it enters Certification Testing. Unlike Validation Testing, the goal is not to “test & fix”. Rather, the goal is to specifically measure the stability and accuracy of the product. The result of this testing, which should include reasonable regression testing, is a final product “health card” which identifies any defects found. The project team and the business sponsor or users then evaluate the test results to determine whether the product should be promoted to production. If it is determined that the product is not sufficiently stable to move to production, it will return to the Development Stage or the Test & Fix stage. There should be no “11th hour” code changes with a forced promotion to production due to the inherent risk.

User Acceptance testing

The purpose of the Acceptance test is to confirm whether the system is ready for operational use. During the Acceptance Test, end users or customers of the system compare the system to its requirements. It may be a requirement that the QA department coordinate and assist the client in this process by creating test cases and tracking any defects reported. User Acceptance Testing should occur in the QA test environment. Note that this Acceptance Test is not a beta test. An Acceptance Test is conducted in a controlled environment while a beta product is released to client environments that are not controlled by the IT group.

Test Result Reporting

The results of the testing efforts are reported to the project team. This report should specify the tests executed, the expected results, and the actual results. The results of the testing should be captured and tracked. Again, this may be accomplished with something as basic as a spreadsheet or a sophisticated defect tracking package.

The results of the test execution are initially reviewed with the Analyst Lead or team. The Analysts are responsible for assigning a “criticality factor” to the defects identified. Business users or beta test customers may be contacted for clarification if necessary. Suggested weightings include:

Value	Types of errors
1: Critical	Key function failure with data corruption No work-around
2: High	Key function failure without data corruption Secondary function failure with data corruption Work-around available Navigation errors
3: Medium	Secondary function failure without data corruption
4: Low	No risk to data or functionality Includes items such as misspelled words, incorrect formatting, etc.

Specific factors considered in assigning a criticality factor will depend upon the system under development. Some considerations may be whether this is a commercial product, a corporate mission critical system, or a support system for a minor business function.

Once the defects have been assigned a criticality factor, the testing results will be reviewed with the team. If this review is part of the Test & Fix cycle, the Technical Lead and Lead Analyst will prioritize the work effort. The Technical Lead will prepare an updated work package or instructions for the development team. While the fix effort should be focused on the more critical defects, lower level defects may be addressed if the component being worked upon contains those errors or if the lower level defects are assigned to more junior members of the development team.

If the testing review is part of the Certification Test cycle, the results will be reviewed with the extended project team including the project sponsor, key users, or customers. The extended team will determine whether the product should be released for User Acceptance Testing or Production. Rarely will a complex system be free of all defects. The purpose of this review is to determine the risk presented by the known defects and to make an informed decision regarding the release of the product. If the product is not released, it may return to further Test & Fix cycles or may be pushed back for a full development iteration.

An additional factor which may be reported is the Defect Discovery Rate. As the test cycles are conducted, a record is kept of the quantity and criticality of defects discovered. Initially, when a product is in its early development iterations, the discovery rate may climb. Over time and development cycles, this discovery rate should decrease indicating improved stability of the product. This is a helpful indication of when a product may be considered eligible for release.

🔗 The Test Results Documents are CM artifacts and eligible for baseline and Change Management control.

Testing Environment

Effective testing is dependent upon having a controlled test environment. In some cases, where multiple products are being tested or multiple operating environments supported, there may be a requirement for multiple test environments. The configuration of these environments must be strictly controlled. Any changes to the operating system, hardware, test tools, networking protocols, etc. will fall under the control of Configuration Management. Each time a test is to be executed in the QA test environment, the systems should be returned to a baseline state. This may only mean resetting the test database. Other times, this may require a fully “cleansed” system – reformatted drives, reinstalled software, etc. The amount of baseline restoration is a consideration of risk management and should be determined by the standards control group within the IT department. If no such body exists, it should be determined by the extended project team.

VI Configuration Management

The Configuration Management Lead (CM) role is the most important role on the project team. The CM Lead is the heart and soul of the project team and without them, the team can not hope to be successful.

The CM Lead owns the following processes:

- Version Control
- Change Management
- System Promotion & Build Management

The Configuration Management manager also participates in processes that are owned by other members of the team. This may include working with:

- Project Manager to identify which project artifacts will be baselined and when
- Infrastructure to promote product to Staging or Production environments

Key artifacts for which the Configuration Management manager is responsible include:

- Code Control Policies and Procedures
- Deployment Document

Role and Responsibilities

Configuration Management is concerned with the control of project artifacts and code once those items have been baselined. The practical purpose for the baseline of an artifact is to acknowledge that the artifact is in a state where consumers of the artifact are executing their processes and preparing their artifacts based upon its state. If the state of the original artifact changes, then the work effort and resulting artifacts based upon the original may also have to change. For example, if code is being developed based upon a design schematic which in turn was based upon a requirements artifact, then a change to the requirements may have a cascading impact on the design and code. Such changes need to be made in a controlled manner.

It is recognized that in this day of shifting requirements and iterative development, change is a constant. The purpose of Configuration Management is not to eliminate or restrict change. Rather it is to ensure the effective communication of change to all stakeholders in the process. Without effective Configuration Management, you cannot have effective rapid application development.

Version Control

The purpose of Version Control is to be able to establish and, if necessary, re-call an artifact or component from a particular point in time. Versioning not only identifies a

temporal state of a particular artifact, but also its relationship to other artifacts at a particular point in time. Instances when this may occur include:

- By Quality Assurance when identifying the version of a product for testing
- By Configuration Management when preparing a package for promotion to the production environment.

The artifacts (project by-products other than application code or database files) and code for each project that will fall under Version Control will be defined at the initiation of the project. There are tools available on the market such as Microsoft's Visual SourceSafe and Rational's ClearCase for tracking and controlling these items. There is a great variance in the capabilities of the various products on the market and care should be taken in selecting one which will best fit the needs of the organization.

The most common project artifacts that will fall under the Version Control process include:

- Scope Document
- Requirements Document(s)
- Design Document(s)
- Source Code including Stored Procedures, Databases, etc.
- Project Plans
- QA Test Plans and Scripts

Generally speaking, only baselined non-code artifacts are stored in the version control database. All application code, database build files, stored procedures, and other code artifacts should be stored regardless of whether they are baselined or not.

Version Control can be relatively simple. It may involve simply tracking a single project artifact through multiple versions. It may also be terribly complex. Examples include tracking multiple versions of products for multiple clients and/or multiple platforms. Tracking multiple versions of products may involve branching and integration of sections of code or components. This requires disciplined procedures, effective tools, and may require significant administrative effort depending upon the capability of the Version Control tools being used.

Change Management

The objective of Change Management is to effectively communicate approved project changes to the team and other project stakeholders. Change Management, sometimes referred to as Configuration Control, is the process by which a change to a baselined item is proposed, evaluated, approved/rejected, prioritized, tracked, scheduled and implemented. This process is owned by the Configuration Management group and is executed by the Change Management Board (CMB). The CMB is comprised of relevant product stakeholders. This group of stakeholders may differ depending upon the project

and the scope of requested changes. There may be an executive level CMB for project changes which impact items such as:

- Functionality
- Performance
- Features
- Cost
- Delivery date

There may be lower level CMB for project changes which do not impact these items but which affect changes which will be realized by members of the team. For example, a minor change in a baselined requirement scenario affecting a design module and a test case which are currently being developed. Another change might be to upgrade an operating system release that will need to occur in the Development and QA environment on a controlled basis. The goal here is not cost or delivery control, but to ensure effective communication.

The CMB should meet on a regularly scheduled basis to review proposed Change Requests (CR) and determine the disposition of the changes. The thoroughness with which the Change Control process will be executed will differ greatly between organizations. Care should be taken to ensure that there is an effective amount of control without creating a disruptive bureaucracy. The Change Control process is most important where there are tight financial controls, large project teams, or cross-project impacts.

Change Request

A Change Request (CR) form is the vehicle used to initiate a change. The Change Request form is filled out and submitted to Configuration Management for processing. The originator should supply, at a minimum, the type of change, a description of the change, a justification (need) statement, an impact statement indicating the cost and benefits of the change, and the identities of the affected artifacts or code components. The request should also note the associated project release (or product version) with which the change is to be associated. The project's CM lead will track all Change Requests from initiation to closure and report the status of the Change Request to the originator, the project team(s), and the CMB.

The Change Management process should always be initiated when a baselined artifact changes. As a result of the artifact baseline procedure, these items will be stored in the Version Control database. The new version of the artifact will be modified and submitted with a notification of the change.

Once the written approvals have been obtained, the changed artifact may be baselined and the change implemented. It is the responsibility of the CM lead, working with the Project Manager, to collect and track approvals as well as the new baseline version of the artifact.

Limit Change, Encourage Evolution

Change is necessary for the healthy development of a product. Many assumptions are challenged during the development process, new information discovered, and required modifications identified. The ability to integrate change effectively into the product development lifecycle is the sign of a mature process. With all that said, however, it must be understood that change *will* have an impact on the project schedule and budget. Even a change which is submitted but not approved still requires project resources to evaluate and consider. And by its very definition, change – the modification of a baseline artifact, will impact multiple players and stake holders on the project.

So how do you allow change and yet keep a project on track? Wherever possible, attempt to push changes into future development cycles. While this is not always practical, it frequently is and will allow you to focus on the regular delivery of a demonstrable product. Since many people find it easier to critique than to create, providing a demonstrable product which they can “experience” will provide a better vehicle for evaluation and feedback. And in many cases, the change requested is additional functionality. Delivering a product with some functionality in a short period of time followed by a more complete version on the next development cycle may be preferable to a product that is constantly delayed by the inclusion of “necessary” changes.

System Promotion & Build Management

The promotion of product from the Development environment into the Quality Assurance environment and, finally, into the Production environment must be disciplined and controlled. This is to ensure that the proper versions of code, stored procedures, etc. are promoted together and that all associated documentation is also catalogued. The actual deployment scripts and instructions will be the responsibility of the project’s Technical Lead but CM will collect and catalogue those items. Using the instructions prepared by the Technical Lead, the CM will compile, or “build”, the system utilizing the proper version of components for the identified release. This activity is carried out in a controlled environment to remove as many system variables as possible and to ensure a reproducible build. Once the system is built, it will be submitted to the Infrastructure team for the actual physical deployment.

It is the responsibility of a project’s CM Lead to collect and tag all baselined items related to a product for a promotion activity. The Project Manager is responsible for delivering and identifying baselined artifacts and it is the Technical Lead’s responsibility to deliver and identify any code or database components. The CM Lead will tag all related items and create a copy in a protected area of the version control database to which only CM has access. Once all the relevant artifacts and code have been tagged and stored, the CM Lead will prepare the Deployment Document. The Deployment Document provides the instructions for promoting the system. It is provided to the Infrastructure team for execution.

🔒 The Deployment Document is a CM artifact and eligible for baseline and Change Management control.

VII. Infrastructure

The Infrastructure Lead (IL) role is the most important role on the project team. The IL Lead is the heart and soul of the project team and without them, the team can not hope to be successful.

The IL Lead owns the following processes:

- Design and implementation of physical environment
- Physical promotion of product
- Physical installation of software tools in QA and Production environments

The Infrastructure Lead also participates in processes that are owned by other members of the team. This may include working with:

- Project Manager to identify capital asset requirements
- Configuration Management to promote product to QA or Production environments

Key artifacts for which the Infrastructure Lead is responsible include:

- Physical infrastructure diagrams
- System configuration documentation

Role and Responsibilities

The role referred to as “Infrastructure” in the OT4D team model is rarely called that in organizations. The responsibilities assigned to this role may be in groups such as:

- Telecommunications
- Network Services
- Production Support
- Operations

Since the names are so inconsistent, the role is merged into a single “lead” in this process. If your organization has this spread across several groups, each would probably have its own lead representative on the project team.

The IL is responsible for the physical systems upon which the software being developed will be executed. This includes the development, quality assurance, and production environments, including computing platforms and data networks, within the organization. This responsibility includes the design, deployment, and support of this infrastructure. The IL is also responsible for the physical installation of the software product as it is

being promoted from the development environment into the QA or Production environments.

The IL needs to be involved with the project from its very earliest stages. Often, new hardware or software may need to be purchased and this process requires a fair amount of lead time in most organizations. Also, ordering telecommunications circuits from the various providers often requires a lead-time of weeks or months. Acquiring telecommunications circuits in overseas locations can sometimes take 6 months or more, even in Europe.

Physical Infrastructure Design

The infrastructure team owns the process of designing the physical maps, including the UML-based Deployment Diagrams, for the project team. During the Design Phase, the IL will work with the Technical Lead to create, and then merge, the physical deployment diagram with the UML-based Component Diagram. This combined diagram will let the team and operations group know how the software will be deployed.

Understanding the issues of bandwidth and circuit access is critical for the project team. Some software designers have little experience with these issues and it is common for developers to assume their network performance in the development environment will be realized in the production environment. It is unfortunately not uncommon for a project team to run through the entire development lifecycle and not realize there are production telecommunication bandwidth issues until the product is released to the user. At this point there is little that the Infrastructure group can do and these teams are often faced with the very unpleasant prospect of having to purchase more bandwidth or re-design the application. Neither alternative is beneficial to one's career.

🔗 The Deployment Diagram is a CM artifact and eligible for baseline and Change Management control.

Product Promotion

The promotion of the product from the development environment to the QA environment and from the QA environment to the Production (or Staging) environment must be tightly controlled. The logical capture and control of the code is executed by the Configuration Management Lead. This process is explained further in the Configuration Management section of this document. However, the actual physical installation of the software is often carried out by the Infrastructure Team.

When the product is ready for promotion, the CM Lead will prepare the final deployment package. This may include scripts and installation routines created by the developers. The final package is turned over to the Infrastructure Lead who oversees the deployment execution. Once the deployment is complete, the IL reports back to the Project Manager or project team the result of the deployment.

Re-Baseline the Environment

The IL is also responsible for the physical act of returning an environment to a baseline state. This is most frequently undertaken when a new build or release of the application is being promoted to the QA environment for testing. At a minimum, this involves removing the previous version of the product and re-installing the test database files. In strictly controlled environments, this may require completely refreshed hard drives and re-built test databases in the test environment.

Less frequently, the IL may be asked to set the Development environment to a previous baseline.

System Configuration Documentation

As part of the responsibility for the physical environments, the Infrastructure team will maintain configuration documentation. These documents will allow for the roll-back to previous baseline configurations if necessary. They will also allow for a rebuild of the system in the event of a catastrophic loss. These documents may be part of an organization's disaster planning effort.

🔒 The System Configuration Document is a CM artifact and eligible for baseline and Change Management control.