CONSTRUX SOFTWARE PRESENTS

SOFTWARE PROJECT MANAGEMENT BOOT CAMP

TOOLBOX

Advancing the Art and
Science of Commercial
Software Engineering

Toolbox Contents

INTRODUCTION

What is a toolbox

Construx finds that the toolbox is very strong metaphor for software development. A software engineering toolbox provides a central knowledge base for acquiring, defining, and disseminating guidance about the software engineering processes and practices. Although particular software tools (e.g., developer environments, revision control systems, automated test suites, project management software) will play a role in the toolbox, **the term "tool" is used in a broader sense to encompass techniques, best practices, templates, checklists, examples, etc**. The techniques employed often have a greater impact than any particular piece of software.

The toolbox provides a mechanism for sharing knowledge about the organization's process assets and how they are used. It accelerates project work by reducing duplication, promoting common usage, and leveraging reuse. The toolbox approach allows the organization to add new tools to their existing way of working without changing everything or throwing away the tools that they already know.

Growing your toolbox

The best toolbox is one that you grow over time. Just like the toolbox you have in the garage, you shouldn't buy a new tool until you need it. A high-level process for growing your toolbox would consist of:

- **Define a simple taxonomy**. This defines how the tools are organized, for example the directory structure in a common repository. The taxonomy allows team members to quickly find tools and easily discuss software engineering issues. A good toolbox taxonomy becomes the framework for your organization to plan, discuss, and execute software project. One trap to avoid is spending too much time trying to define taxonomy. It is more important to have a common, well-understood organization than to have one that is "perfect."
- **Bootstrap the toolbox.** Populate the toolbox with an initial set of tools. Bootstrapping consists of **harvesting** tools that the teams already have defined and identifying and **filling in critical gaps**. Harvesting leverages investments your company has already made and promotes buy-in. It also sends a clear message that management is not trying to impose a one-size-fits-all set of practices. You should then identify and fill critical gaps by creating or acquiring the necessary tools.
- **Continuous improvement**. The toolbox should be the first place they look for a tool. In many cases, the team will be able select the appropriate tool from the toolbox. However, the team will face situations where existing tools do not meet their needs. Teams should approach this by first seeing if a tool could be modified for the use. If not, they can create a new tool. In both cases, the team should put the tool back in the toolbox for future consideration.

Using the Toolbox

The toolbox approach does not mandate a particular lifecycle or methodology. It specifically recognizes that the one-size-fits-all approach is fundamentally flawed. Instead, it acknowledges that the team is ultimately responsible for identifying the tools that they will use on their projects.

The toolbox provides the project members the framework in which to select the right-sized tool and optimize it for their particular needs. The tools can be applied separately or in concert with one another. They can be mixed and matched with other tools to support different teams and objectives.

Construx's Toolbox

Most of the examples in this document are based on artifacts in Construx's CxOne toolbox. CxOne is an agile framework of templates, checklists, patterns, and other tools that provide relief from software development pain. CxOne provides practical software engineering support to project teams. This vision is realized through efficient execution of proven techniques.

CxOne is based on industry best practices, Steve McConnell's works, Construx's real-world project experience, and lessons learned while working with Construx's clients. CxOne does not prescribe a single approach to software development; it provides a foundation that allows projects and organizations to quickly improve how they develop software. CxOne does not replace your existing practices; it builds on them. CxOne's document-based approach provides maximum flexibility for teams to deploy new and improved practices.

For more information, see Construx's CxOne web page.

Project Charter Template

Authority	
Agent(s)	
Completion criteria	
Resources	
Constraints	
Priorities	
Assumptions	

Project Charter Checklist

The project charter is the vehicle for driving project inception and winning approval to execute a project. It defines the top-level goals, responsibilities, and context for the project.

1. The project charter is based on the charter template.
2. The charter document complies with documentation standards.
3. The authority that is chartering the project is clearly identified.
4. The identified authority is recognized by all participants as having the authority to charter this project.
5. All true agents in the project have been identified. This includes anyone who is responsible for seeing that the project is carried out successfully.
6. All identified agents are really responsible, and they are being delegated authority, in the project.
7. All true completion criteria have been clearly identified.
8. Every completion criterion is really applicable to the project.
9. The completion criteria are as objectively measurable as possible.
10. The completion criteria do not confuse the problem with the solution. In other words, the completion criteria are problems to be solved, not solutions to be implemented.
11. All appropriate criteria for canceling the project are included.
12. All resources being made available are clearly identified.
13. In chartering, the authority is committing to making the stated resources available to the agents.
14. All significant constraints are clearly identified.
15. All stated constraints are really constraints on this project.
16. All assumptions are clearly identified and relevant.

Sample Vision and Goals Statements

Introduction

The project Vision is a brief summary of the goals and scope of the project, i.e., what the project will accomplish and what is its focus. It should give some understanding as to why the project exists and call out guiding principals for the project as they relate to strategic objectives of the organization. A good vision provides guidance both about what a project is, and what it is not. The vision may be qualitative and does not need to be measurable, but it should be traceable to specific project goals that are measurable.

Goals are the desired results of the project from a business perspective, i.e., what is the project accomplishing. Describe "what" is to be accomplished, not "how". Remember the goals must be SMART (Specific, Measurable, Attainable, Relevant, and Timely). Try to be as quantitative and precise as possible.

Normally the Vision and Goals are defined by the project sponsor (authority) and is included in the Project Charter. However, if one doesn't already exist, you should create one for you own protection.

The following links are three sample vision and goal statements. The first two samples use the "Business Requirement + Strategy" format. The third example uses the Planguage format. One of the key differences is it uses a scale (target, stretch, failure) rather than single point measures.

The samples are from:

- Defect Tracking System. This sample was extracted from a prior version of Construx's Requirements Boot Camp seminar. The labs in that seminar were based on building a Defect Tracking System.
- Data Warehouse Implementation. The second sample came from the Charter Document that Construx helped a client develop. To date, they had invested approximately $750k in implementing data warehousing and Online Analytical Processor (OLAP) tools. It was generally accepted that OLAP's potential for supporting information-based decisions was largely untapped. The client wanted to maximize this investment by using OLAP in making business decisions
- Embedded Device. The third example is from a case study used in Construx's Quality Planning Seminar. The labs in that seminar were based on building an embedded digital light processor.

From Defect Tracking System Case Study

The following has been extracted from a prior version of Construx's Requirements Boot Camp seminar. The labs in that seminar were based on building a Defect Tracking System.

Vision

Increase the productivity/efficiency of our software development processes by using quantitative defect metrics.

To support this vision, we will:

- Create a defect tracking system that will be used consistently by all projects to collect and report on defect data, while still allowing teams to define a workflow that fits their development processes.
- Produce quantitative defect metrics to improve our visibility of the status of active projects and to create explicit release criteria.
- Maintain historical information about our defect totals and trends to better predict and manage future projects.

Measurable Goals

G1 Increase productivity/efficiency

- G1.1 At least 75% of the total defects detected are identified during upstream reviews (i.e. requirements and design reviews)
- G1.2 The average cost (effort and duration) to fix a defect decreases by an order of magnitude
- G1.3 The total number of Severity A (i.e. show stoppers) identified as design or coding defects are reduced on each project

G2 Used consistently throughout the organization

- G2.1 Team members use the defect tracking system on an almost daily basis throughout the project and especially during a defect correction cycle
- G2.2 Software engineers are able to use the tools without guidance within two weeks of being introduced to them
- G2.3 Once trained, software engineers can switch to another project's defect tracking workflow with no more than one hour of guidance

G3 Fits team's development processes

- G3.1 Project teams can teams can setup a workflow in under an hour
- G3.2 Teams can save their workflow definitions and use them to setup future projects
- G3.4 Teams integrate defect tracking with other systems/data they use to track projects
- G3.3 Teams export data into the Excel spreadsheets they use for analysis, reporting and charting

G4 Improved visibility of the status of active projects

- G4.1 Daily use of reports to track progress of defect correction
- G4.2 The expected vs. actual defect trend rates is tracked on all projects
- G4.3 Defect reports are used to approve releasing systems
- G4.4 No reports of defects that have been inadvertently dropped

G5 Better predict and manage future projects

Historical data is used on all projects to:

- G5.1 Predict rework rates and drive estimates for corrective activity effort
- G5.2 Predict expected defect trend rates and drive test and quality planning
- G5.3 Create explicit release criteria

From an Data Warehouse Implementation Project

This example came from the Charter Document we created for a local client. To date, they had invested approximately $750k in implementing data warehousing and Online Analytical Processor (OLAP) tools. It was generally accepted that OLAP's potential for supporting information-based decisions was largely untapped. The client wanted to maximize this investment by using OLAP in making business decisions

Vision

Improve the timeliness and quality of decisions by providing the decision maker with access to pertinent information.

To support this vision, we will

- Promote general acceptance of the data warehouse Online Analytical Processor (OLAP) as a tool for information-based decisions at all management levels
- Provide access and support to anyone who wants to use OLAP to support their decision needs.

Measurable Goals

G1: Promote use and acceptance

Provide successful examples of OLAP solving the following business problems:

G1.1 <Specific problem was defined>

G1.2 <Several more>

G2: Increase trust in information available

G2.1 Executive Council request OLAP reports 90% of the time for their presentations

G2.2 100% of the sites are either polled or accounted for in an exception report within 3 days of month end

G3: Provide Training to OLAP Users

G3.1 Training is provided for three levels of OLAP users <Levels were defined>

G3.2 90% of training participants pass standard assessment tests confirming Level I, II, III abilities

G4: Provide Support to OLAP Users

G4.1 Support experts are trained to Level II training specifications and are available to Level I users

G4.2 90% of questions regarding use of OLAP tool are resolved within 1 business day

G4.3 90% of technical issues are resolved within 1 business day

From an Embedded Device Case Study

This example came from a case study in Construx's Quality Planning Seminar.

Vision: Improve our customer satisfaction by increasing battery life through correct screen lighting of our handheld device

Goals

Customer Satisfaction

Gist	How satisfied the customer is with our handheld device
Measure	Mean Satisfaction Score from our customer satisfaction survey given six months after purchase
Target	10% improvement from current satisfaction survey results
Stretch	15% improvement
Failure	< 8% improvement

Usability

Gist	How easily can an end customer adjust the key lighting parameters
Measure	Time required to train a typical end customer to adjust the standard three parameters
Target	Two minutes or less of training
Stretch	Zero minutes
Failure	> 10 minutes

Reliability

Gist	Does not cause the screen to go blank
Measure	Number of use scenarios that generate blank screens;
Target	Zero in the top 80% of use scenarios
Stretch	Zero in top 95% scenarios
Failure	> 1 in the top 50% scenarios

Functionality

Gist	Adjust the light display as necessary
Measure	% of correct adjustments to changing light conditions;
Target	95% of 50 correct adjustments in an automated 10 light level test
Stretch	99%
Failure	85%
Measure	Number of severity-1 (subsystem does not work), severity-2 (subsystem mostly works), or severity-3 (subsystem works but data gets corrupted) defects released to production

Target	No sev-1, sev-2, or sev-3 level defects
Stretch	No sev-1, sev-2, or sev-3; failure
Failure	No sev-1, > two sev-2 with workarounds, > three sev-3 with workarounds
Measure	Number of sev-4 defects (inconvenient) per 1,000 LOC be released to production
Target	2
Stretch	0
Failure	>10

Tips for Documenting the Iron Triangle

This document illustrates a couple techniques for documenting the iron triangle constraints.

The iron triangle is often used as a metaphor for the Project Management's triple constraints: Schedule, Functionality, and Resources. The fundamental planning principle is the triangle must remain in balance. One way to put the principle in words is:

In a given amount of time, with a given team, you can only produce a set amount of functionality.

Pre Fixe Menu Metaphor

This metaphor is based on restaurants that offer Pre Fixe menus (i.e. For $25 you get your choice of one appetizer, one entrée, and one desert). The columns communicate increasing degrees of freedom.

	Fix	Aim	Adjust
Schedule	Must be ready for Feb's Trade Show		
Resources			10 Engineers Can add up to 5 contractors
Functionality		Complete infrastructure layer Sufficient business layer to: • Exercise the infrastructure • Be included as an alpha release to a "new customer" • Dumb UI for Expert User (eventual goal is a smart UI)	

You can use different labels such as "Mandatory," "Optimize," and "Negotiable." The point is to use terms that makes sense for your organization.

Target Zone Metaphor

Another approach is using the target zone. It communicates more degrees of freedom. The key thing to remember when using the target zone is the columns do not reflect different scenarios (i.e. worst case, expected case, and best case). The target zone communicates where tradeoffs can be made and still meet the project objectives.

For example, an acceptable scenario could be delivering the target functionality in January by adding the 5 contractors, and delivering the target functionality.

	Priority	Accept	Target	Stretch
Schedule	1	March	February	January
Resources	3	10 Engineers Up to 5 contractors	10 engineers	Use only 8 engineers
Functionality	2	Complete infrastructure layer Sufficient business layer to: • Exercise the infrastructure • Be included as an alpha release to a "new customer" • Configured using note-pad (same as current product)	Complete infrastructure layer Sufficient business layer to: • Exercise the infrastructure • Be included as an alpha release to a "new customer" • Dumb UI for Expert User (eventual goal is a smart UI)	Complete infrastructure layer Sufficient business layer to: • Exercise the infrastructure • Be included as an beta release to a "new customer" • Smart UI for an intermediate user

You can substitute "Tolerate" for "Accept." It really doesn't change the meaning, but some business people prefer it that way.

The target zone concept also works great for your project's quality requirements which also often have conflicting goals (e.g. an embedded device requirements for battery life conflicts with the requirements for bright displays). In this case, the target zone communicates where trade offs can be made and still result in an acceptable product.

Project Plan Template

The project plan is the top-level controlling artifact of a project. It defines how the project will achieve the goals defined in the project charter. It defines planning for executing more along with managerial and technical processes.

A project plan may be a single document or a hierarchical collection of artifacts contained under a root document. A project plan may cover a single project, or it may cover a collection of sub-projects.

Project Plan Lifecycle

A project plan is normally started as part of the chartering process, and an initial draft may be completed along with the charter. Although the project plan may evolve along with the project, it should always define enough detail to allow upcoming and ongoing work to be efficiently planned and executed. Normally a project plan should define details as far as possible into the future, preferably covering the entire life of the project, even if future details are less precise or detailed than near-term ones.

Project Plan Template

Introduction

This section describes any necessary background or context information for the project. Often much of the background and context information for the project is captured by the project's charter. This section should reference the project charter and elaborate on it if appropriate.

Overview

This section provides an entry point for understanding the project and the environment it is taking place in. This section may provide more detail on items described in the project charter, but should simply refer to the charter if there is nothing worth calling out in greater detail.

If the project plan is long, this section may also be a useful place to summarize key points related to project planning.

If there will be global strategies applied to the project that do not fit within other planning areas, or span multiple areas, they can be described here. An example of a global strategy would be taking a conservative or aggressive approach to project risk tolerance, e.g., relating to decisions about expected values for estimates, maturity of processes and technology, etc.

Deliverables

Lists the artifacts that will be delivered externally by the project. The deliverables represent the ultimate tangible manifestation of the project's software requirements. The level of detail used here should normally be at a summary level. Since this information is often captured in detail elsewhere, it may often be appropriate to simply describe the "XYZ system" as the project deliverable, along with any other deliverables interesting to external stakeholders that they would consider separate from the software (e.g., training to use the software).

If the project is part of a contractual arrangement, this section should mirror the contractual deliverables, or better yet, simply refer to the contract.

These top-level project management deliverables should be traceable to the business schedule section and the requirements for the system the project is building.

Business Schedule

This section can define a simple business schedule or summarize a more complex one. The business schedule is the top-level schedule defining project milestones and how project deliverables and activities map onto calendar time. Usually a detailed business schedule is developed as a separate project artifact.

If a project is schedule driven, the business schedule defines targets the project is managing towards. If the project is feature or budget driven, the schedule represents estimates of how the schedule will work while the project is being managed to other drivers (like feature or budget). Since scheduling is dealt with in detail in other areas of project planning, this section should simply be a brief recap focused on schedule targets, estimates, or constraints that are critical to the project's success.

Resources

This optional section summarizes the use of effort/cost or other limited resources the project will use. Normally it would only be included in the introduction if resource usage was a major driver identified in the project charter. If the resources are the primary driver for the project these section describes targets to be managed towards. If the project is schedule or feature driven this section represents estimates of what resources will be expended.

Resources, especially staff effort (which is a normally a direct proxy for cost), are dealt with in depth in other planning areas, so this section should simply be a brief recap focused on resource targets, estimates, or constraints that are critical to project's success.

Assumptions and Constraints

Lists the formal or informal assumptions and constraints that could impact the overall planning of the project. Do not duplicate information dealt with in other global project planning materials unless there is value to elaborating here. For example, if time constraints are dealt with in the business schedule, there isn't a need to repeat the same information here.

Often the information in the project charter is sufficient, in which case this section can simply refer to the charter.

Assumptions and constraints that pertain to narrow areas of the project should be captured in any planning artifacts closer to the area in question. For example, assumptions and constraints specific to testing should be covered in the test plan.

Risks and Assets

Lists the formal or informal risks and assets that could impact project planning. This section can normally refer to the risks and assets documented as part of the project charter. Any risks and assets elaborated here should cover items that are likely to be a major ongoing factor for project planning. Specific, timely risks are managed through separate explicit risk management materials.

Terms and Acronyms

If appropriate list any project definitions for terms and acronyms introduced by this plan. Often this section can be skipped in favor of a global project terms and acronyms document.

Reference Materials

If appropriate, list any external materials referenced by the plan or which it was based upon.

Project Lifecycle

This section should describe the complete, customized lifecycle for the project. This includes defining the abstract modeling of project work, and the definition of precise phases and stages relevant to the project.

Lifecycle Model

Normally a well known lifecycle will be selected for a project, or a set of lifecycle elements combined. The lifecycle elements should be identified, defined if references are not available, and the reasons for selection discussed. Customizations lifecycles should be noted, along with any combinations or hybridizations of multiple lifecycles. Ideally reasons for not selecting alternatives should be discussed.

As an example, a project using a straight staged delivery lifecycle could simply state that this was the case, and explain the rationale for selection.

Lifecycle Map

This section takes the lifecycle model defined in the previous section and maps it onto the specifics of this project. This includes defining the phases that make up the project's lifetime from inception to closure and major stages of development. This lifecycle description should map onto the deliverables, business schedule, and resource usage described in the introduction. The lifecycle map should not simply repeat information contained elsewhere, it should provide a view of the project that is centered around the lifecycle, as opposed to the schedule, requirements, etc.

As an example, the project in the previous section using a straight staged delivery lifecycle could define specific up-front phases (e.g., Phase 1 is planning and requirements, Phase 2 is architecture and high-level design, Phase 3 is delivery), and the define the delivery stages that are expected at this time (e.g., Stage 1, initial feature, Stage 2 data entry, etc.).

For iterative lifecycles, the lifecycle map will likely not be able to map out the entire lifecycle in detail. When this is the case, make the map as precise as far in to the future as possible, and then use placeholders to represent lifecycle phases and stages that will be defined in more detail later.

Management Structure

This section defines how the project will be structured to make use of resources.

Project Organization

External Interfaces

Described the important external interfaces for the project. This could include the sponsor, other external stakeholders, departments, managers, vendors, project reviewers, auditors, etc.

If this project is part of a tree or web of sub-projects, the projects it interfaces with should be identified.

Roles and Responsibilities
Project roles should be abstractions that define roles to fulfill project activities. This section does not cover the assignment of individuals. Rather it sets up the abstract roles and responsibilities that are then assigned to individuals in the next section.

Staffing
The staffing plan is often contained in a separate document if it is expected to be modified frequently. On smaller projects, one person may play many or all roles.

Internal Structure
If appropriate describe the internal structure of the project team. This section is normally omitted unless an org-chart of the project team is not self-evident from the roles and responsibilities description.

Risk Management

The purpose of this section of project planning is to describe how risks will be managed on the project. CxOne defines two types of risk management. Implicit risk management is the management of risks intrinsic to all engineering activities. Explicit risk management is the application of risk management practices to identify, prioritize, and mitigate risks.

On many projects it may be appropriate to simply state that risks will be managed intrinsically and through the use of a top risks list in the project status report. Large, complex, or highly constrained projects may require significantly more formal risks management techniques, e.g., assigning a risk officer.

Implicit
In this section discuss any conscious risk mitigation that occurred as part of global project planning. A common example is the selection of the project lifecycle, the level of formality in project processes, or a strategy to use mature vs. newer technologies or practices.

This section should be used to summarize key project decisions made to address risks identified as part of the project chartering. It is definitely not intended as an exhaustive list of the risk analysis associated with every project decision.

Explicit
This section describes the explicit risk management practices that will be used to manage ongoing risks on the project. This includes risk techniques like a top 10 risks list, having weekly risk reviews, assigning a risk officer, etc.

Issue Management

This section covers how issues will be managed on a project. Issues are defined as unplanned work or other things that crop up which must be addressed for the project to succeed. Issue management processes work well when built around database driven tools, but can be managed with a simple list of issues.

Communication

This section should dictate how, when, and what information will flow between project participants and stakeholders. Communication is an intrinsic part of running a sound project and overlaps with other areas of the project plan, especially the tracking area.

Many projects can just call out the major communication devices and processes here, e.g., weekly executive status reports. Some projects may require a more detailed or formal description of communication needs, or define a schedule of communication artifacts, events, or milestones that need to occur during the project.

Startup

Any significant issues a project needs to deal with to get up and running they should be dealt with here. Many projects in existing organizations can skip this section or refer to organizational processes for starting up projects. This is because startup issues are often implied by staffing and other planning.

Closeout

As with Startup, this section is only needed if closing down the project is a large scale exercise or unique to normal organizational processes.

Planning and Control

This section should define the precise planning, tracking, and control mechanisms that will be used on the project.

Estimates

This section normally will simply refer to a separate project estimate artifact.

Estimation Process

Describe the estimation process used to create the initial project estimate(s). Outline how and when the estimate will be refined throughout the project. This information will sometimes be easier to manage if it is also included in the project estimate artifacts.

Resource Identification

List the resources available to the project including information on available staff, time, budget, materials, etc. When appropriate, describe the resource obtainment and phase out timelines. If appropriate, this section can detail any gaps between the available resources and the resources necessary to complete the project.

Effort

The focus of this section is to describe the staff/effort resources that are available to the project. Describe the staffing on the project as an abstracted model capturing important specializations, effort profile over time, total expected effort, etc. If appropriate, include best case, worst case, and nominal case.

Time

Describe the available calendar time for the project. If appropriate, include best case, worst case, and nominal case. This section can often just refer to the business schedule. If the business schedule is based on an expected case or is a balance of competing constraints and

uncertainty, this section can be used to summarize the raw information that went into creating the expected case.

Cost
If not directly related to effort, describe the available budget for the project. If appropriate, include best case, worst case, and nominal case.

External Resources
If appropriate detail external resources that need to be actively managed for the project to succeed.

Materials and Other Resources
If appropriate list any materials or resources that the project will need to manage. This can be used as catch-all bucket for items that should be mentioned but don't fit in the other categories.

Resource Allocation

Describe the allocation of the identified resources to meet the project goals. Consider the following issues, activities, and artifacts:

- A work breakdown structure of the projects activities
- A work plan defining effort estimates and rough scheduling information
- Detailed schedules defining dependencies and fine-grain scheduling information
- Resource allocation to different functions during the project (e.g., engineering, content, quality, management, etc.)
- Internal and external dependencies
- Resource/staff availability throughout the project
- Staffing profiles throughout the project
- Resource procurement throughout the project
- Availability of budget and budget dependencies

The following sub-sections often simply contain pointers to other project artifacts:

Work Breakdown Structure
The WBS can be captured either as part of a work plan or as a dedicated artifact. This section can simply point to the artifact that captures the WBS explicitly or implicitly. This section can discuss any context surrounding the creation or management of the WBS that is not captured elsewhere.

The WBS decomposes all project work into work packages, providing a view that allows project work to be managed. Iteration is often necessary for a WBS, because decomposition can only go so far at early stages of the project. It is assumed that the WBS will be refined as the project progresses.

Work Plan
Work plans are normally a separate artifact that is referenced from this section. This section can describe the goals of the project's work plan and its relation to project schedules and corrective activity management (CAM). Some projects will not have schedules but rely on the work plan and CAM to handle scheduling of activities.

Scheduling
Describe or reference the schedule(s) for the project. There are usually several different schedules of varying levels of detail that describe the allocation of work packages. CxOne describes the broad categories of business schedule and detailed schedule. Most schedules are stored in separate documents or specialized schedule tools.

Some projects may not use detailed scheduling, relying instead on a work plan and CAM to manage scheduling of project tasks. If this is the case it should be noted here.

Budget
If appropriate describe the allocation of budget resources to the project. Many projects don't require a dedicated budget plan because the budget is directly related to effort and staffing.

Tracking and Control

Describe how project cost, schedule, quality, and functionality will be tracked and controlled throughout the project. Consider the following elements:

- Work plan based tracking
- Effort tracking or time accounting
- Earned Value Management (EVM)
- Corrective Activity Management (CAM)
- Detailed schedule tracking
- Informal management techniques (management by walking around)
- Collaborative team techniques (collaborative construction, Extreme Programming)
- Status reports (content, structure, and frequency)
- Project website
- Audit mechanisms

This section will normally delegate quality tracking and control details to the quality plan and functionality tracking to the CM or requirements management plan.

This section and the communication section will compliment each other (e.g., status reports and project websites have applications in both areas). Information should not be needlessly duplicated, but there may be issues for the same item that are unique to its use.

Technical Process

This section should describe the engineering and technical environment in which the project is occurring.

Software Engineering

This section covers the engineering methodologies, practices, techniques, and tools to be used on the project. Since many of these will be defined as part of organizational practices or as part of industry or third-party support (e.g., CxOne), this section can often just reference the appropriate resource being leveraged.

Also, since management processes are dealt with by other sections in this plan, and CM, quality, and testing processes are normally addressed in their dedicated plans, those areas do not normally need to be addressed.

Beyond references to external materials, this section should cover any deltas or unique aspects of this project in regards to engineering methodologies including requirements and design methodologies, practices, and notations, construction standards, documentation standards, system integration procedures, and so on.

The template suggests an organization based on the following CKAs, whose processes are normally addressed directly by other project documentation:

Requirements
The strategies, techniques, etc. used to elicit, analyze, specify, and manage all types of requirements.

Design
Design levels, approaches, strategies, methodologies, techniques, notations, etc. Often some of this will be touched upon in the architecture document.

Construction
The use of standard practices, styles, templates, idioms, conventions, etc.

Technology

This section covers the use of technology on the project, including hardware and software platforms, tools, components, etc.

Technology Management
Describe how technology will be evaluated, selected, and managed for the life of the project. If the technology has been mandated as part of the project's constraints, note that here.

Development Environment
The computing system environment including all hardware and software elements necessary to provide different elements of the development environment. This would include software tools including design tools, source code control, time accounting, compiler or IDE, debugging aids, defect tracking, and so on.

Be sure to identify non-standard requirements for non-engineer stakeholders. For example, external stakeholders who need access to the CAM database.

System Technology
Describe the hardware and software platforms and environment necessary to support the software system resulting from the project.

Infrastructure
Discuss any work necessary to create or configure the development environment(s).

Environment

The development environments in which the project will occur. For example, corporate headquarters, off site, distributed work environment, etc.

Artifacts

This section is also known as a document list. An organizational process like CxOne may negate the need for detailed artifact lists if all project artifacts are fairly standard and their delivery is described in other places. If a detailed list of configuration items (CIs) is created in the CM plan, this section can reference the appropriate areas of that list.

Use this section if there are significant non-standard project artifacts, or if this is the best location to summarize all project artifacts in one place.

Supporting Plans

This section normally references detailed supporting plans necessary to round out project planning. Most projects will have separate CM, quality, and test plans. The other plans are optional, but most projects should capture some information on these issues. In some cases instead of a separate plan, a short paragraph in this section would provide enough coverage for the product.

Configuration Management

Describes how the project identifies and controls the project artifacts including change control, builds, and release processes.

Quality

Describes how the project ensures it meets the quality goals including reviews, audits, etc.

Testing

Describes how the project will execute and track the testing activities.

Deployment

Describes how project will deploy and support the product.

Integration

Describes integration between sub-projects, tasks, releases, etc.

System/Product Acceptance

Describes the customer acceptance criteria, activities, and management processes relating to accepting system or product deliveries.

Operations

Describes the mechanism for ongoing operation of the product.

Maintenance

If this is a maintenance project or it will have a maintenance phase, the maintenance plan can deal with issues specific and unique to building on top of the existing system.

Procurement

Mechanism for procurement of materials or sub-contracted services on the project.

Staff Development

Describes any significant training, teambuilding, or other staff development activities.

Project Plan Checklist

Use this checklist to review your project plan.

1. The plan is based on the proper template.
2. The plan document complies with documentation standards.
3. All assumptions on which the plan is based are adequately identified and described.
4. All work defined in the complete project plan is necessary to satisfy the charter, control risks, and/or maximize assets.
5. All work defined in complete project plan is sufficient to satisfy the charter, control risks, and/or maximize assets.
6. All available resources are adequately identified and all resources identified as being available are actually available.
7. The available resources are adequately and appropriately allocated to the defined work (i.e., not over- or under-utilized).
8. All derived/induced risks and assets have been identified and accounted for in the complete project plan.
9. Uncertainties ("TBD"s) in the project plan are both obviously identified and appropriate.
10. Where appropriate, all project plan components (deliverables, practices, resource allocations, etc.) are explicitly justified in light of the charter, and Risk and Asset assessments.

CxOne Risk and Asset Pattern

You can use the risks in this pattern as the catalyst for identifying risks and assets on your projects. Often you can cut and paste items from this pattern to form the basis for your project's risk management identification.

Schedule Creation

Id	Description	Risk	Asset
sc1	Schedule, resources, and product definition have all been dictated by the customer or upper management and are not in balance		
sc2	Schedule is optimistic, "best case," rather than realistic, "expected case"		
sc3	Schedule omits necessary tasks		
sc4	Schedule was based on the use of specific team members, but those team members were not available		
sc5	Cannot build a product of the size specified in the time allocated		
sc6	Product is larger than estimated (in lines of code, function points, or percentage of previous project's size)		
sc7	Effort is greater than estimated (per line of code, function point, module, etc.)		
sc8	Re-estimation in response to schedule slips is overly optimistic or ignores project history		
sc9	Excessive schedule pressure reduces productivity		
sc10	Target date is moved up with no corresponding adjustment to the product scope or available resources		
sc11	A delay in one task causes cascading delays in dependent tasks		
sc12	Unfamiliar areas of the product take more time than expected to design and implement		

Organization and Management

Id	Description	Risk	Asset
om1	Project lacks an effective top-management sponsor		
om2	Project languishes too long in fuzzy front end		
om3	Layoffs and cutbacks reduce team's capacity		
om4	Management or marketing insists on technical decisions that lengthen the schedule		
om5	Inefficient team structure reduces productivity		

Id	Description	Risk	Asset
om6	Management review/decision cycle is slower than expected		
om7	Budget cuts upset project plans		
om8	Management makes decisions that reduce the development team's motivation		
om9	Non-technical third-party tasks take longer than expected (budget approval, equipment purchase approval, legal reviews, security clearances, etc.)		
om10	Planning is too poor to support the desired development speed		
om11	Project plans are abandoned under pressure, resulting in chaotic, inefficient development		
om12	Management places more emphasis on heroics than accurate status reporting, which undercuts its ability to detect and correct problems		

Development environment

Id	Description	Risk	Asset
de1	Facilities are not available on time		
de2	Facilities are available but inadequate (e.g., no phones, network wiring, furniture, office supplies, etc.)		
de3	Facilities are crowded, noisy, or disruptive		
de4	Development tools are not in place by the desired time		
de5	Development tools do not work as expected; developers need time to create workarounds or to switch to new tools		
de6	Development tools are not chosen based on their technical merits, and do not provide the planned productivity		

End-users

Id	Description	Risk	Asset
eu1	End-user insists on new requirements		
eu2	End-user ultimately finds product to be unsatisfactory, requiring redesign and rework		
eu3	End-user does not buy into the project and consequently does not provide needed support		
eu4	End-user input is not solicited, so product ultimately fails to meet user expectations and must be reworked		

Customer

Id	Description	Risk	Asset
cu1	Customer insists on new requirements		
cu2	Customer review/decision cycles for plans, prototypes, and specifications are slower than expected		
cu3	Customer will not participate in review cycles for plans, prototypes, and specifications or is incapable of doing so—resulting in unstable requirements and time-consuming changes		
cu4	Customer communication time (e.g., time to answer requirements-clarification questions) is slower than expected		
cu5	Customer insists on technical decisions that lengthen the schedule		
cu6	Customer micro-manages the development process, resulting in slower progress than planned		
cu7	Customer-furnished components are a poor match for the product under development, resulting in extra design and integration work		
cu8	Customer-furnished components are poor quality, resulting in extra testing, design, and integration work and in extra customer-relationship management		
cu9	Customer-mandated support tools and environments are incompatible, have poor performance, or have inadequate functionality, resulting in reduced productivity		
cu10	Customer will not accept the software as delivered even though it meets all specifications		
cu11	Customer has expectations for development speed that developers cannot meet		

Contractors

Id	Description	Risk	Asset
co1	Contractor does not deliver components when promised		
co2	Contractor delivers components of unacceptably low quality, and time must be added to improve quality		
co3	Contractor does not buy into the project and consequently does not provide the level of performance needed		

Requirements

Id	Description	Risk	Asset
req1	Requirements have been baselined but continue to change		

Id	Description	Risk	Asset
req2	Requirements are poorly defined, and further definition expands the scope of the project		
req3	Additional requirements are added		
req4	Vaguely specified areas of the product are more time-consuming than expected		

Product

Id	Description	Risk	Asset
prd1	Error-prone modules require more testing, design, and implementation work than expected		
prd2	Unacceptably low quality requires more testing, design, and implementation work to correct than expected		
prd3	Development of the wrong software functions requires redesign and implementation		
prd4	Development of the wrong user interface results in redesign and implementation		
prd5	Development of extra software functions that are not required (gold-plating) extends the schedule		
prd6	Meeting product's size or speed constraints requires more time than expected, including time for redesign and re-implementation		
prd7	Strict requirements for compatibility with existing system require more testing, design, and implementation than expected		
prd8	Requirements for interfacing with other systems, other complex systems, or other systems that are not under the team's control result in unforeseen design, implementation, and testing		
prd9	Pushing the computer science state-of-the-art in one or more areas lengthens the schedule unpredictably		
prd10	Requirement to operate under multiple operating systems takes longer to satisfy than expected		
prd11	Operation in an unfamiliar or unproved software environment causes unforeseen problems		
prd12	Operation in an unfamiliar or unproved hardware environment causes unforeseen problems		
prd13	Development of a kind of component that is brand new to the organization takes longer than expected		
prd14	Dependency on a technology that is still under development lengthens the schedule		

External environment

Id	Description	Risk	Asset
ex1	Product depends on government regulations, which change unexpectedly		
ex2	Product depends on draft technical standards, which change unexpectedly		

Personnel

Id	Description	Risk	Asset
per1	Hiring takes longer than expected		
per2	Task prerequisites (e.g., training, completion of other projects, acquisition of work permit) cannot be completed on time		
per3	Poor relationships between developers and management slow decision making and follow through		
per4	Team members do not buy into the project and consequently does not provide the level of performance needed		
per5	Low motivation and morale reduce productivity		
per6	Lack of needed specialization increases defects and rework		
per7	Personnel need extra time to learn unfamiliar software tools or environment		
per8	Personnel need extra time to learn unfamiliar hardware environment		
per9	Personnel need extra time to learn unfamiliar programming language		
per10	Contract personnel leave before project is complete		
per11	Permanent employees leave before project is complete		
per12	New development personnel are added late in the project, and additional training and communications overhead reduces existing team members' effectiveness		
per13	Team members do not work together efficiently		
per14	Conflicts between team members result in poor communication, poor designs, interface errors, and extra rework		
per15	Problem team members are not removed from the team, damaging overall team motivation		
per16	The personnel most qualified to work on the project are not available for the project		
per17	The personnel most qualified to work on the project are available for the project but are not used for political or other reasons		
per18	Personnel with critical skills needed for the project cannot be found		

Id	Description	Risk	Asset
per19	Key personnel are available only part time		
per20	Not enough personnel are available for the project		
per21	People's assignments do not match their strengths		
per22	Personnel work slower than expected		
per23	Sabotage by project management results in inefficient scheduling and ineffective planning		
per24	Sabotage by technical personnel results in lost work or poor quality and requires rework		

Design and Implementation

Id	Description	Risk	Asset
des1	Overly simple design fails to address major issues and leads to redesign and re-implementation		
des2	Overly complicated design requires unnecessary and unproductive implementation overhead		
des3	Inappropriate design leads to redesign and re-implementation		
des4	Use of unfamiliar methodology results in extra training time and in rework to fix first-time misuses of the methodology		
des5	Product is implemented in a low level language (e.g., assembler), and productivity is lower than expected		
des6	Necessary functionality cannot be implemented using the selected code or class libraries; developers must switch to new libraries or custom-build the necessary functionality		
des7	Code or class libraries have poor quality, causing extra testing, defect correction, and rework		
des8	Schedule savings from productivity enhancing tools are overestimated		
des9	Components developed separately cannot be integrated easily, requiring redesign and rework		

Process

Id	Description	Risk	Asset
pro1	Amount of paperwork results in slower progress than expected		
pro2	Inaccurate progress tracking results in not knowing the project is behind schedule until late in the project		
pro3	Upstream quality-assurance activities are shortchanged, resulting in time-consuming rework downstream		

Id	Description	Risk	Asset
pro4	Inaccurate quality tracking results in not knowing about quality problems that affect the schedule until late in the project		
pro5	Too little formality (lack of adherence to software policies and standards) results in miscommunications, quality problems, and rework		
pro6	Too much formality (bureaucratic adherence to software policies and standards) results in unnecessary, time-consuming overhead		
pro7	Management-level progress reporting takes more developer time than expected		
pro8	Half-hearted risk management fails to detect major project risks		
pro9	Software project risk management takes more time than expected		

Risk Templates

Risk Register

Rank	Risk Name	Enabling Event	Triggering Event	Dissipating Event	Severity	Probability	Status / Notes

Top 10 Risk List

Rank	Risk	Last Report	Weeks on List	Next Action
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Risk Management Plan Template

Risk Management Plan

Risk Identification

Id		**Status**		**Priority**	
Submitter		**Owner**		**Verifier**	
Title					
Enabling Event					
Trigger Event					
Dissipating Event					
Related Risks					
Expected Close Date		**Actual Closed Date**		**How Resolved**	TBD

Risk Analysis

Initial Exposure

Date Identified	Probability Factor	Impact Rating	Risk Exposure	Recommended Action

Current Exposure

Date Last Reviewed	Probability Factor	Impact Rating	Risk Exposure	Recommended Action

Risk Control

State	New	**Next Action**		**Next Action Date**	

Risk Management Plan		
Action to control the risk	**Assigned to:**	**Target Date**

Risk Notes		
Date	**Author**	**Note**

Risk and Asset List Checklist

Use this checklist to review the risks and assets you identified on your projects.

1. The risk and asset analysis is based on the proper template.
2. The risk and asset analysis document complies with documentation standards.
3. No significant project risks or assets are missing from the assessment.
4. Each identified risk or asset is really important to the project. Only items that impact the project should be mentioned.
5. The root cause of the risks and assets is beyond control of the project team.
6. Each identified risk or asset is quantified in terms of an estimated probability of impact.
7. Each estimated probability is reasonable.
8. Each identified risk or asset is quantified in terms of an estimated degree of impact (severity).
9. Each estimated degree of impact is reasonable.
10. It is clear from the risk and asset assessment what risks and assets need to be managed for the project.

Software Project WBS Pattern

WBS ID	Work Package Name
P	**Initiate Project**
P1	Startup
P2	Create Planning Infrastructure
P3	**Acceptance Planning**
P3.1	Define Acceptance Criteria
P3.2	Create Acceptance Tests
P4	**Project Planning**
P4.1	Create Project Plan
P4.2	Create CM Plan
P4.3	Create QA Plan
P4.4	Create Test Plan
P5	**Setup PM Infrastructure**
P5.1	Setup Project Web Site
P5.2	Setup CAM Database
P5.3	Setup Tracking Tools
P5.4	Setup Quality Metric Reporting
P6	**Setup Development Environment**
P6.1	Setup Servers
P6.2	Setup Developer Workstations
P6.3	Setup Revision Control System
P6.4	Setup Build Framework
P6.5	Setup Test Framework
E	**Execute Project**
UM	**Implement User Management**
UM1	Complete Requirements
UM2	Complete High-Level Design
UM3	**Implement User Management Use Cases**
UM3.1	Add User
UM3.2	View User
UM3.3	Edit User
UM3.4	Delete User
UM4	**Implement User Access Use Cases**
UM4.1	Logon User
UM4.2	Logoff User
UM4.3	List Online Users
UM4.4	Cancel User
UM4.5	Lockout Users

WBS ID	Work Package Name
PA	**Implement <another product area set>**
PA1	Complete Requirements
PA2	Complete High-Level Design
PA3	**Implement <feature set> Use Cases**
PA3.1	<use case 1>
PA3.2	<use case 2>
PA3.3	<use case 3>
PA3.4	<use case 4>
PA4	**Implement <feature set> Use Cases**
PA4.1	etc ….

X	**Closeout Project**
X	**Conduct Acceptance Tests**
XM1	Complete Test
XM2	Complete Reports
XM3	**Conduct Project Retrospective**
XM3.1	Gather Data
XM3.2	Conduct Interviews
XM3.3	Prepare Report
XM3.4	Present Findings
M	**Control Project**
M1	**Milestone 1**
M1.1	Do Project Management
M1.2	Do Technical Management
M1.3	Maintain Infrastructure
M1.4	Do General Quality Control
M1.5	Do General Rework
M1.6	Do Integration Testing
M1.7	Do Builds and Releases
M2	**Milestone 2**
M2.1	Do Project Management
M2.2	Do Technical Management
M2.3	Maintain Infrastructure
M2.4	Do General Quality Control
M2.5	Do General Rework
M2.6	Do Integration Testing
M2.7	Do Builds and Releases

WBS ID	Work Package Name
M3	**Milestone 3**
M3.1	Do Project Management
M3.2	Do Technical Management
M3.3	Maintain Infrastructure
M3.4	Do General Quality Control
M3.5	Do General Rework
M3.6	Do Integration Testing
M3.7	Do Builds and Releases

Sample Rolling Wave Plans

This sample illustrates a Milestone Implementation Roadmap and a Milestone Delivery Plan. In an actual project, each of these would be separate documents.

The samples are based on a fictitious project that handles sales transactions and payments. It assumes the system is being created from scratch, so the team has to put into place the project infrastructure (revision control, automated builds, test, etc) and product services (error handling, menu harness, access management, etc).

The sample implementation roadmap includes only the first few milestones – and actual roadmap would include all the milestones. It is also a good practice to start each milestone on a new page.

The sample milestone delivery plan is for the second milestone.

Implementation Roadmap

Purpose

The Ajax Project Implementation Roadmap defines the high-level path through the project and the major dependencies. It is used to facilitate communication, planning and control across team. It defines primary reporting and tracking vehicle for communicating status outside the development teams.

Planning guidelines

Defining the milestones was based on the following guidelines:

- **Define completion criteria for each milestone**. The completion criteria will be used to guide the selection of work packages included in the milestone.
- **Maintain release quality.** Bring Ajax to near-release quality on a frequent basis. This helps keep the quality high and reduces the risks of uncovering a fatal defect late in the process.
- **Provide frequent stable releases.** We need to provide the Core and the Extended teams stable, frequent releases in order for them to progress on their work.
- **Use Rolling Wave Planning.** Include updates to the plans and estimates as more information is uncovered.
- **Anticipate mid-course changes.** Allow changes to future milestones to respond to market changes.
- **Include both the Core Development & Extended Teams.** The milestone definition and completion criteria must present a complete picture of what needs to be done across both the Core and Extended Teams.

General Strategy

The Ajax Project will be using incremental development. The project has been divided up into milestones that provide the details of when work will be accomplished. Milestones will be a time box activity, typically lasting one month.

The general strategy for this plan is to deliver functional capability in a pipeline fashion as illustrated in the *Ajax Project Plans – Lifecycle Description* document.

The Ajax Sales System will be implemented in the following order:

- Project infrastructure
- Basic application framework
- High risk items
- Features in priority order

Reference Materials

- Ajax Project Plans
- Ajax Project Lifecycle
- Ajax Work Breakdown Structure

Definitions and Acronyms

See the Ajax Terms and Acronyms document.

Global Completion Criteria

This section references the following completion criteria that apply to all milestones.

C1 **Work associated with prior milestones have been completed**

C1.1 Defects found during system testing or reviews in prior milestones have been addressed (e.g. resolved, rejected, or deferred).

C1.2 Data entities and attributes introduced or changed during the prior milestone have been included in the data conversion and migration procedures.

C1.3 Any help text created by the business team has been included into the build.

C2 **Work associated with the current milestone has been completed**

C2.1 All work needed to satisfy the completion criteria for the current milestone has been completed or a decision has been made on what to do about the uncompleted items.

C2.2 All work packages allocated to the current milestone has been closed or a decision has been made on what to do about the open items.

C3 **Team is ready for future work**

C3.1 Requirements and design for features scheduled to be completed in the next stage are defined to a level sufficient to proceed.

C3.2 Lessons learned have been captured for process improvement

C3.3 Project infrastructure (tools and procedures) has been updated as needed.

C3.4 The Milestone Delivery Plan and Work Plan for the next stage have been completed.

C3.5 Resources have been identified and made available for work.

C3.6 The milestone review has been conducted and the sponsor has given approval to proceed.

C4 **All work products are updated and controlled.**

C4.1 All project artifacts (plans, requirement documents, designs, code, etc) have been updated as necessary and have been checked into the revision control system.

C4.2 Any release associated with the milestone has been created using the automated build system.

Milestone Specific Completion Criteria

M1 – Project Infrastructure

Objective
Define development processes and procedures that will ensure the on-going integrity and success of the Ajax project. Effort will be concentrated on: automating software construction and testing practices, establishing internal and external project communication forums, and completing the work of defining the Ajax project.

Completion Criteria
M1.1 The project plans have been completed and approved.

M1.2 The project infrastructure (hardware, software, and/or procedures) needed for the following items have been implemented and tested. This includes the revision control system, automated build system, automated test system, corrective activity management system, and project web portals.

M1.3 A preliminary evaluation of the available 3rd Party Payment Processor systems has been conducted and has identified the top 3 candidates for further evaluation. The criteria and plan for selecting the 3rd Party Payment Processor has been created..

Target Environment
Development

M2 – Basic application framework

Objective
Put into place the basic underlying services that is common across the system. Primary focus of this milestone will be on implementing the public interfaces. Services will be limited to basic functionality. The services will be extended to full functionality during later milestones.

Completion Criteria

M2.1 Public interfaces to the Menu Harness, License Manager, Audit Manger, Event Manager, Help Harness have been implemented and tested.

M2.3 Developer Manual for using the services has been provided to the teams.

M2.4 The 3Rd Party Payment Processor has been selected

M2.5 A paper UI prototype for the Sales Subsystem has been created

Target Environments
Development, Integrated Test

M3 – Cash Sales

Objective
Exercise the basic underlying services that are common across the system by recording a simple cash sale. Demonstrate feasibility of connecting the UI layer to the database layer. Additionally, the audit and error handling systems will be enhanced.

Completion Criteria

M3.1 A simple cash sale transaction can be recorded in the database.

M3.2 The transaction has been recorded in the audit log.

M3.3 The error handling routine has been exercised.

Target Environment
Development, Integrated Test

M4 – And so on for the rest of the milestones

Milestone Delivery Plan

Executive Summary

Overview	This milestone deliverable plan is for Milestone 2 of the Ajax project. The purpose of this milestone is to deliver the basic underlying services that are common across the system. Primary focus of this milestone will be on implementing the public interfaces. Services will be limited to basic functionality. The services will be extended to full functionality during later milestones.
Scope	Public interfaces to the Menu Harness, License Manager, Audit Manger, Event Manager, Help Harness Developer Manual for using the services Selection of the 3Rd Party Payment Processor Paper UI prototype for the Sales Subsystem
Schedule	August 1 – 31, 2005

Milestone Delivery Plan

Scope

Deliverables included in this milestone

Deliverables	Description	Acceptance Criteria
Public Interfaces	Menu Harness – The structure to build menus and toolbars. License manager – Public interface to determine if a feature is licensed or not. At this point, the licensee manager will just be stub that can be used to manually test the interface. The rules engine will be developed in later milestones Access Manager – Public interface to determine if a user has access to the feature and data. At this point, the access manager will just be stub that can be used to manually test the interface. The rules engine will be developed in later milestones. Audit Manager – Public interface to pass audit requests and data. At this oint the audit manager will just write data to a text file. The business logic to write and manage audit logs will be developed in a later milestone. Event Manager – Public interface to pass event details. At this point the audit manager will just write data to a text file.The business logic to write and manage event logs will be developed in a later Help Harness – Public interface to insert calls to the help system. Error Handling – Basic error handling services to include information and warning dialog boxes.	Unit test approved by the project and team technical leads
Developer Manual	A light-weight developer manual defining how to use the services.	Inspection -- approved by the project and team technical leads
3rd Party Payment Processor	The 3rd Party Payment Processor has been selected.	Inspection – Approved by the project and team technical leads.
	A proposed contract has been provided to the legal department.	Inspection – Approved by the Project Manager
	The payment processor has been integrated into the project plans and designs.	Inspection – Approved by the team leads
UI Prototype	A paper prototype developed and presented to targeted customers.	Inspection – Approved by the UI Design Lead

Assumptions

This plan's scope and scheduled are based on the assumptions documented in this section. Deviations from these assumptions may require adjustment to the plan.

- All teams will take part in creating the designs for the infrastructure services.
- All critical stakeholders have sufficient time to prepare for and participate in project activities.

Shared Deliverables

The following table describes those tasks and deliverables where coordination between the teams is required to meet the Milestone Completion date. The schedule depends on a timely completion of the following:

Shared Deliverable Schedule

Description	Resource	Expected Effort	Date Required
Approval of M2 requirements	Requirements Lead	5-8 hours	Aug 1, 2005
Participate in a 1-2 day Design workshop to address high profile issues	Subject Matter Experts, Design Lead	8 – 16 hours per person	Aug 5, 2005
Review Developer Manual	Technical Leads	2 – 4 hours per person	Aug 19, 2005
Check in completed M2 work	Build Manager		Aug 26, 2005
Participate in requirements related & design related tasks for M3 requirements (This is to set an expectation. It is very possible we will not need this much time, or that it will be required of only selected people)	Requirements Lead, Subject Matter Experts	8 - 16 hours per person	Aug 29, 2005
Participate in M2 Review and M3 Planning	Team Leads	8 hours per person	August 30 – 31, 2005

Issue Resolution

In addition to the tasks and deliverables stated above, issues may be discovered during this plan's execution and assigned to team resources. Timely resolution of these issues is essential to the successful delivery of this milestone and the entire project. For each issue, team leads will determine a reasonable and timely resolution date. For reference, the following table provides guidelines for resolving issues in a timely manner:

Issue resolution timeliness guidelines

Priority	Description	Expected Frequency within M2	Expected Resolution Timeline
Critical	Blocking forward progress	0-2	1-2 business days
Very High	Impeding milestone delivery	0-10	2-3 business days
High	Impeding timely completion of scheduled tasks	10-50	3-5 business days

Train Metaphor

Adopting a metaphor for releases can be helpful to communicate release information throughout the organization. Many companies have found metaphors simplify discussions around the features included in a release by providing a language that all parties can relate to. It also lessens the political dimension that frequently results when discussing capacity using cost or staff hours.

The following "train" metaphor is as a suggestion to consider for discussing releases. A train represents a release; the train seats represent the resource capacity; and the passengers represent features. This type of metaphor can work well within a defined portfolio management process.

The train metaphor works on several levels:

- A train always leaves on a predefined timetable. It can be delayed somewhat while passengers are loading, but not much. The releases are deployed on a predefined timetable. They can be delayed somewhat while functionality is being implemented, but not much.
- A train has a preset capacity. Once the seats are full, remaining passengers must wait for the next train. If the train will pick up passengers at another station, the seats must be reserved. The releases have a preset capacity, as they are resource constrained. Once the release capacity has been reached, new features must wait for the next release.
- The train capacity is defined by how many passengers the train can take at one time. If you need to carry more passengers, you must add another car to the train. The release capacity is defined by how much the software engineering team can successfully analyze, design, construct, test and deploy in a given timeframe. If you need to implement more functionality in a given timeframe, you must increase the size of your software engineering team.
- The train's crew assignment is carefully planned. Switching crew members between stations is very costly and delays the schedule. If the train crew member has to unexpectedly switch between trains, both trains' schedules are impacted. The software engineering team's capacity must factor in anticipated switching.
- The software engineering team can determine what their staff month capacity is within a specified timeframe. They can then translate that into a sizing unit called "seats", which represents a specified number of staff months. Features are then estimated in how many "train seats" are required. Estimates are provided in ranges, from best to worst case. The range between the best and worst case can be narrowed as requirements and design are completed, providing more precise estimates and better understanding of what will be included.

The following table displays how you could use the train metaphor to represent a release during project planning. In this example there are 18 "seats" available for the project.

Prioritized Feature List Seats Required =>	Best Case	Expected Case	Worst Case
Reserved capacity for change requests	3	4	5
Feature A	1	1	2
Feature B (infrastructure for future release)	2	3	5
Feature C	3	5	5
Feature D	1	1	3
Feature E	3	4	4
Feature F	3	4	8
Feature G	5	6	7
Feature H	1	2	2
Feature I	2	3	4

Legend

	Within capacity
	Within capacity assuming switching does not exceed planned
	Additional capacity required

The table illustrates a number of key concepts:

- The cut-line describes a risk profile associated with how much functionality will be committed to; engineering's can only provide a "guarantee" of Features A, B, and C. Nominal planning should assume Features A-F, but that may need to be adjusted.
- The program anticipates needing to respond to changing user needs or late arriving features. The team has reserved capacity for change requests during the release.
- The team will be working on Feature B that will not be included in the release. This often occurs when a feature will take more than one release to fully develop.
- The spread of the capacity lines (e.g. between best and worst case) demonstrates the cone of uncertainty as applied to date-driven projects.
- The estimates, and therefore the capacity lines, factor in the lost productivity based on the anticipated amount of switching due to customer support and working on other projects. The features below the dashed line (highlighted in yellow) are those that will have to be dropped if the team has to switch more than anticipated.

The train metaphor can be extended to represent projects during strategic panning. Instead of using seats and features to represent capacity and demand, you would use a coarser unit called train cars (e.g. the engineering department capacity is 10 train cars and project one needs 1 – 2 cars). The coarser unit reinforces the Cone of Uncertainty concepts discussed in the seminar.

Lifecycle Models

This section describes the following Software Development Lifecycles:

- Waterfall
- Staged Delivery
- Incremental Delivery
- Design to Schedule
- Evolutionary Delivery
- Evolutionary Prototyping

Waterfall

Waterfall is a lifecycle model in which software is developed in orderly sequence of steps from the initial concept through system testing.

Main Benefits	Helps minimize planning because you can do all the planning up front
Keys to Success	Having stable product definition.
When to Use	Waterfall works best for with stable requirements. You may consider using when building a well-defined maintenance release or porting a product to a new platform.
Main Risks	The main risk associated with waterfall is the difficulty of fully specifying the requirements at the beginning of the project.

Overview

The waterfall is the most common and most misused lifecycle. In the waterfall model the project progresses through an orderly sequence of steps from the initial concept through system testing. The project holds a review at the end of each phase to determine whether it is ready to advance to the next phase. The waterfall model is document driven, that is the main work products that are carried from phase to phase are documents.

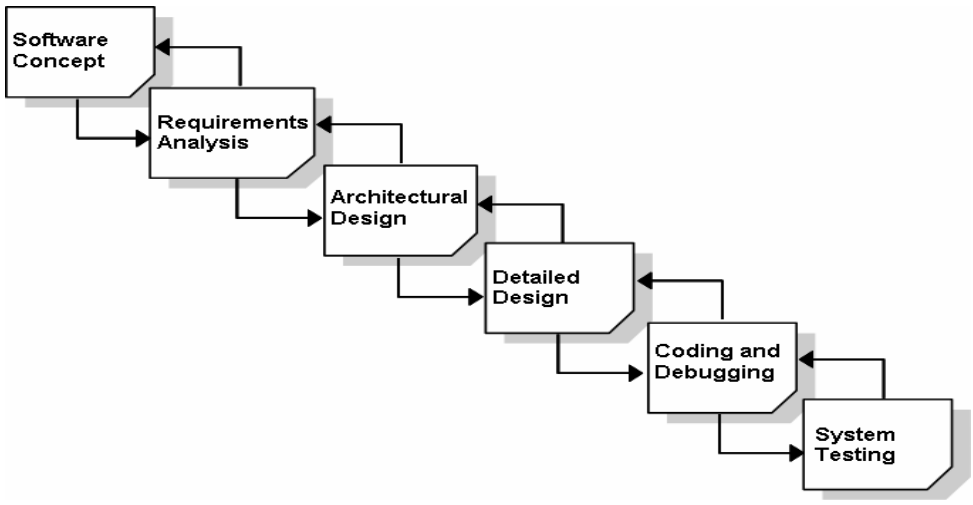


Figure C1: Waterfall Delivery Lifecycle

Interactions with other Best Practices

Although it has many problems, it serves as the basis for other, more effective lifecycles.

Staged Delivery

Staged delivery is a lifecycle model in which software is developed in stages, usually with the most important capabilities being developed first.

Main Benefits	Increased visibility of progress and support for more frequent and predictable product releases.
Keys to Success	Ensuring the product architecture is solid and supports more than one possible product direction, defining delivery stages carefully, releasing the first stage as early as possible, releasing the stages in their order of importance, and planning to handle customer change requests.
When to Use	Staged delivery works best for well-understood systems, very large projects when your customers are eager to begin using a relatively small portion of the product's functionality, and for systems in which you can develop useful subsets of the product independently. If you can't figure out how to break the delivery of your product up into stages, staged delivery is not the right approach for you.
Main Risks	The main risk associated with staged delivery is the risk of feature creep. When customers begin to use the first release of your product they are likely to want to change what has been planned for the other releases.

Overview

In staged delivery the most important capabilities are generally developed first. Staged delivery doesn't reduce the time needed to build a software product, but it substantially reduces the risks involved in building one. It provides tangible signs of progress that are visible to the customer and useful to management in assessing project status.

The staged delivery lifecycle progresses through the waterfall model steps of defining the software concept, analyzing requirements, and creating an architectural design. It then proceeds to do detailed design, coding, debugging, and testing within each stage, creating a releasable product at the end of each stage.

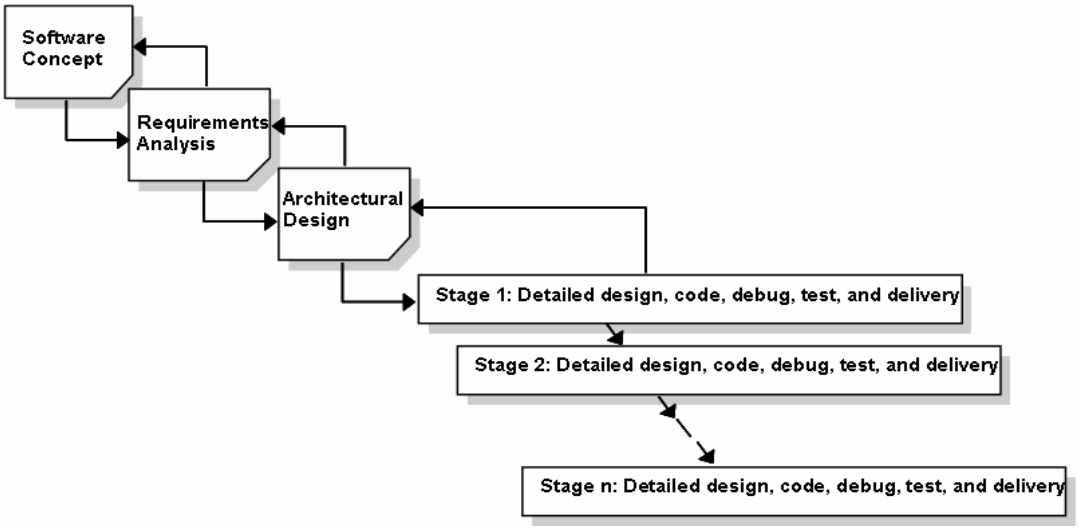


Figure C2: Staged Delivery Lifecycle

Interactions with other Best Practices

Although there are a few similarities, staged delivery is not a form of prototyping. Prototyping is exploratory, and staged delivery is not. Its goal is to make progress visible or to put useful software into the customers' hands more quickly. Unlike prototyping, you know the end result when you begin the process.

If staged delivery provides less flexibility than you need, you can probably use evolutionary delivery or evolutionary prototyping instead. If you know the general nature of the system you're building but still have doubts about significant aspects of it, don't use staged delivery.

Staged delivery combines well with miniature milestones. By the time you get to each stage, you should know enough about what you're building to map the milestones out in detail.

Success at developing a set of staged deliveries depends on designing a family of programs. The more you follow that design practice, the better able you'll be to avoid disaster if the requirements turn out to be less stable than you thought.

Modifications

Incremental Development – System is built in stages and bought to near-release quality at the end of each stage. It is not released outside of development. This results in frequent integration which increases the opportunity for finding defects early.

Design to Schedule – Same as above, except the development ends when time runs out. As a result, low priority features may be left out of the release.

Evolutionary Delivery

Evolutionary delivery is a lifecycle model that supports delivery of selected portions of the software earlier than would otherwise be possible, but it does not necessarily deliver the final software package any faster.

Main Benefits	Increases visibility of progress, supports mid-course corrections, and supports for more frequent and predictable product releases.
Keys to Success	Ensure the system's core is carefully defined, develop an architecture that supports more than one possible product direction, and schedule the delivery stages to release functionality beginning with the "most certain".
When to Use	Evolutionary delivery works best for systems where the core functionality is well-understood, but a fair amount of uncertainty exists in other areas such as the user interface, the secondary feature set, etc.
Main Risks	The main risks associated with evolutionary delivery are the risk of feature creep and diminished project control.

Overview

Evolutionary delivery is a lifecycle model that strikes a balance between staged delivery's control and prototyping's flexibility. It provides some ability to change product direction mid-course in response to customer requests. Evolutionary delivery has been used successfully on in-house business software and shrink-wrap software. Used thoughtfully, it can lead to improved product quality, reduced code size, and more even distribution of development and testing resources.

To use evolutionary delivery the core functionality of the system must be well understood. If your system is poorly understood and you expect a lot of surprises, including surprises that are likely to affect the system in fundamental ways, you will be better off using prototyping.

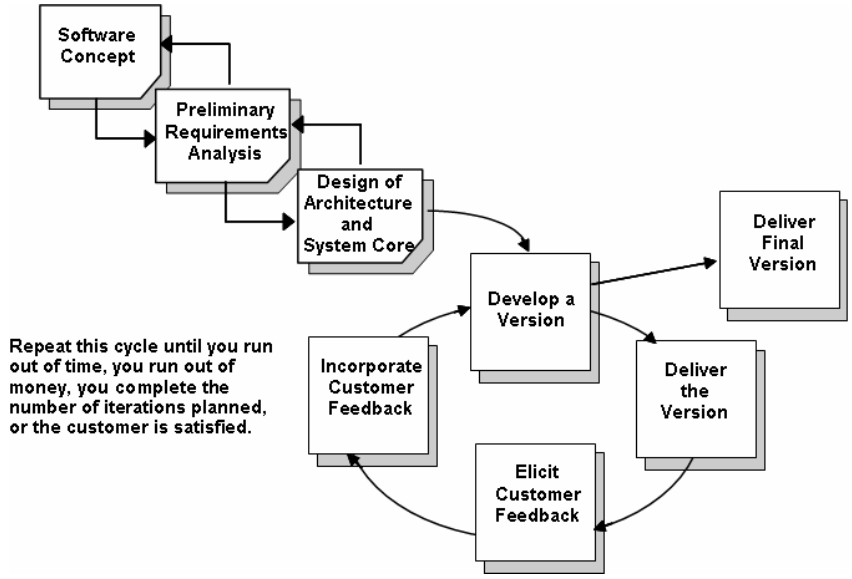


Figure C3: Evolutionary Delivery

Interactions with other Best Practices

Success of evolutionary delivery depends on the effective use of designing for change. Evolutionary delivery invites changes, and your system needs to be set up to accommodate them.

Evolutionary Prototyping

Evolutionary prototyping is a lifecycle model in which the system is developed in increments so that it can readily be modified in response to end-user and customer feedback.

Main Benefits	The ability to address risk early in the project, early feedback on whether the final system will be acceptance, and visible progress throughout the project.
Keys to Success	Using experienced developers, managing schedule and budget expectations, and managing the prototyping activity itself.
When to Use	Evolutionary prototyping works best when the customer is uncertain about what they want at the outset and their input must be solicited throughout the project to ascertain requirements.
Main Risks	The main risks associated with evolutionary prototyping are unrealistic schedule and budget expectations, inefficient use of prototyping, unrealistic system performance expectations, and poor design.

Overview

In evolutionary prototyping the system concept is developed as you move through the project. You begin by developing the most visible aspects of the system. You demonstrate that part of the system to the customer, and then continue to develop the prototype based on the feedback you receive. At some point, you and the customer agree that the prototype is "good enough," and you release the prototype as the final product.

It is probably best suited to business systems in which developers can have frequent, informal interactions with end-users. But it is also well suited to commercial, shrink-wrap, and systems projects as long as you can get end-users involved. The user interaction for these kinds of projects will generally need to be more structured and formal.

If evolutionary prototyping provides less control than you need or you already know fundamentally what you want the system to do, you can use evolutionary delivery or staged delivery instead.

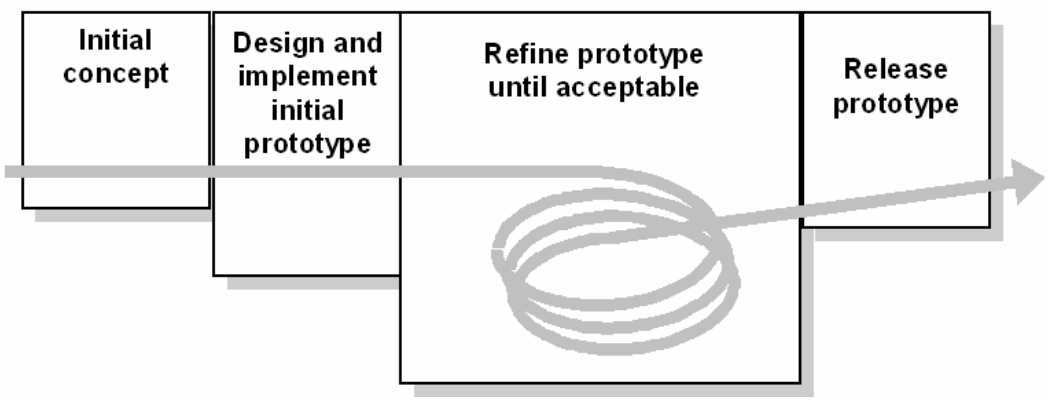


Figure C4: Evolutionary Prototyping

Interactions with other Best Practices

Prototyping is an effective remedy for feature-creep when used with other practices.

Prototyping is also an effective defect-removal practice when combined with other practices. A combination of reviews, inspections, and testing produce a defect-removal strategy with the highest defect removal efficacy, lowest cost and shortest schedule. Adding prototyping to the mix produces a defect-removal strategy with the highest cumulative defect-removal efficacy (Pfleeger 1994a).

Recommended Sources Of Information

For more information on the topic discussed in this section, see the following references:

McConnell, Steve. *Rapid Development*. Redmond WA: Microsoft Press. 1996.

____, *CxOne Process Framework*, Construx Software, 2002

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

SAMPLE DETAIL LIFECYCLE MODEL

The detail lifecycle model describes the activities (e.g. requirements, design, construction, test, etc) that will occur during the lifecycle in greater depth. It clearly defines and communicates how the team will work with each other. The model identifies:

- The hard-constraints that must be complied with (e.g. Sabanes-Oxley, Product Stage Gates, etc)
- Which activities and techniques will be used on the project and what are the completion criteria
- When activities occur and the amount (if any) between iterations. (E.g. our iterations will be one-month time boxes. During the first week will be planning & design, middle weeks would be construction, and the last week will be integration and stabilization.)

The model becomes an important reference for your team. But the real value is having the team work through the decisions needed to create their model. It provides the avenue for the team to "learn together" by providing the opportunity for them to explore the different concepts, decide which ones make sense, and define what it means to them.

The following sample is based on a modified stage delivery lifecycle. It is provided for illustration purposes only—**your model will be different**, reflecting your unique needs and culture.

Modified Stage Delivery using a Pipeline Pattern

Overview

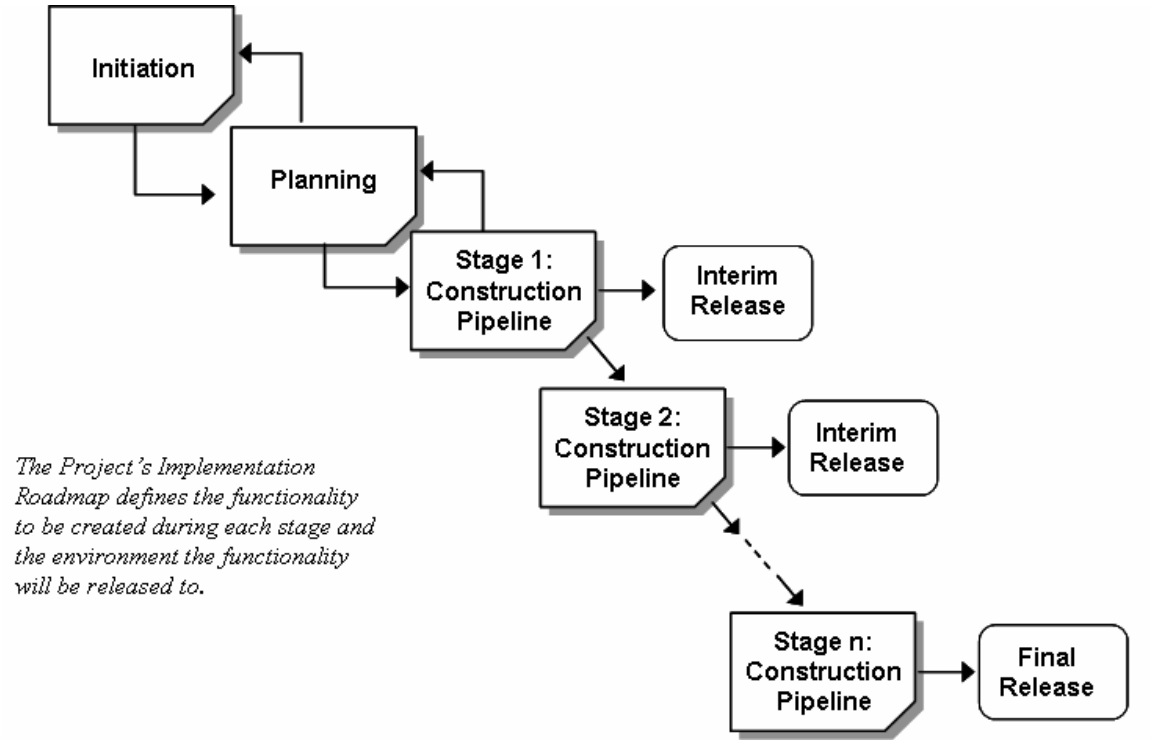
This lifecycle is a hybrid between the Stage Delivery and Evolutionary Prototyping lifecycles. It strikes a balance between staged delivery's control and prototyping's flexibility. The most significant features of this pattern are:

- Deciding upfront the project scopes, high-level requirements, and order of implementation.
- Use rolling wave planning – a progressive detailing of the project plan by providing the details of the work to be done in the current project phase; but also providing some preliminary description of work to be done in later project phases.
- Delaying detail requirement definition of each feature until the construction is about to start.
- Using a construction pipeline consisting of overlapping stages. This takes into account different groups of people are involved in creating the requirements, constructing the features, and testing the features.
- Maintaining near-release quality throughout the construction pipeline. Near-release quality is defined to be a state where the organization could decide to stop feature

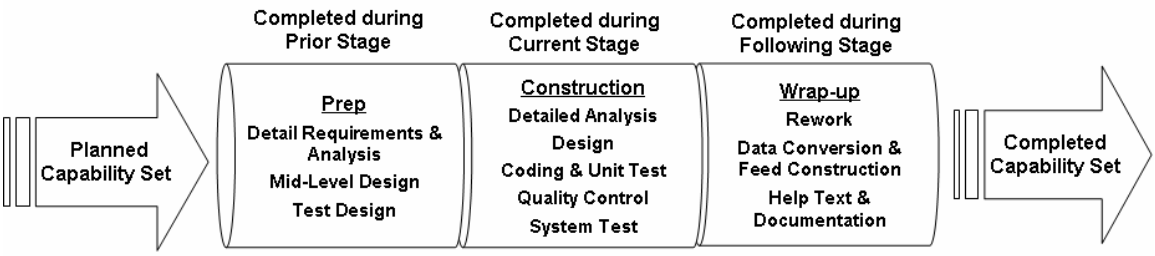
development and the team could bring the system to release quality in no-more than one additional stage.

The following figures depict the model, construction pipeline, and the relationships to rolling wave planning. Additional descriptions of the activities that are included in each element following the figures.

Lifecycle Model

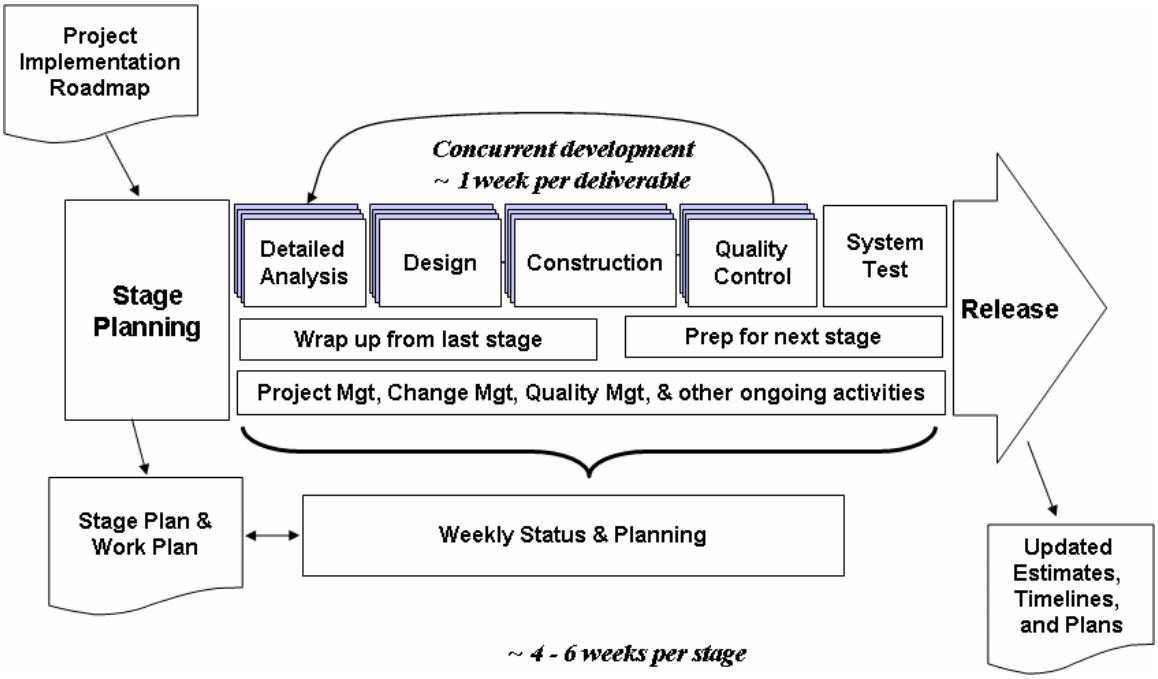


Construction Pipeline



Capability Set – The set of features and other capabilities (e.g. improved infrastructure) needed to satisfy the completion criteria defined in the project's Implementation Roadmap.

Incorporating Rolling Wave Planning



Activity Descriptions

Planning Activities

Activity	Includes activities such as
Initiation	Defining the software concept. Conducting initial risk and asset assessments. Creating the project plan for the planning stage. Creating and approving the charter that defines the boundaries of the project.

Activity	Includes activities such as
Planning	This will be unique for each project. Specific activities will be defined in the project plan for the planning stage. Typical activities include: • Selecting the appropriate-weight management and technical processes that will be used on the project. • Setting up the project infrastructure (e.g. revision control, requirements management, automated test and build environments). • Defining the high-level requirements and high-level architecture. • Creating the project estimates and business schedule. • Creating the implementation roadmap. • Validating the business case. • Identifying team members. • Priming the construction pipeline by completing the prep-work for the 1st construction stage. • Conducting the Planning Checkpoint Review

Construction Pipeline Activities – Prep for next stage

Activity	Includes activities such as
Detailed Requirements and Analysis	Requirements discussions, meetings, etc UI Prototyping Create (update) the Software Requirements Specification Reviews of requirements related artifacts Immediate rework of any identified requirements issues *Note: These activities will primary focus on defining the requirements needed for the next stage*

Activity	Includes activities such as
Mid-Level Design	Create (update) the high-level design to include the business objects and classes Create UI designs. Revised the Architecture as needed Reviews of design and architecture related artifacts Immediate rework of any identified issues Updating the estimates and plans from a construction perspective (e.g. adding/deleting work packages) *Note: These activities will primary focus on defining the high-level design needed for the next stage.*
Test Design	Create (update) the high-level design for test needed to confirm the features have been implemented. Revise the Test Plan as needed. Reviews of test related artifacts Immediate rework of any identified issues Updating the estimates and plans from a testing perspective (e.g. adding/deleting work packages) *Note: These activities will primary focus on defining the tests needed for the next stage*

Construction Pipeline Activities – Construction

Activity	Includes activities such as
Detail Analysis	Requirements analysis, planning, meetings, discussions, research, etc. Creation of requirements artifacts Completing UI prototyping and design Reviews of requirements related artifacts Immediate rework of any identified requirements issues
Design	Design analysis, planning, meetings, discussions, research, etc. Creation of high-level and low-level design artifacts (external & internal designs) Test case development Reviews of design related artifacts Immediate rework of any identified design issues

Activity	Includes activities such as
Construction -- Product Features	Complete any remaining low-level or UI design activities Construction analysis, planning, meetings, discussions, research, etc. Creating product code Unit test creation and automated test coding Formal and informal reviews, inspections, and desk checks Construction testing (includes ad hoc bench testing, unit testing, and build testing) Debugging of issues found during construction testing Create configuration data or infrastructure as needed Other activities that directly support construction Immediate rework of any identified issues
Construction -- Test Cases	Designing and creating test cases Creating scripts used to load test data System test creation and automated test coding Formal and informal reviews, inspections, and desk checks Debugging of test issues found during testing Other activities that directly support test case construction Immediate rework of any identified issues
Quality Control	All post-construction testing performed by someone other than the person who developed the work-product. Any reviews of deliverables or items that were not reviewed earlier Verifying all reviews and tests have been accomplished Putting help text and/or documents created by the users into the revision control system and the release Promoting code to the build Immediate debugging / rework of any identified issues
System Test	Feature testing Integration testing Data conversion and data feed testing Other testing as required by the Test Plan (e.g. regression, load, stress) Immediate rework of any identified issues

Activity	Includes activities such as
Release	Applying release labels. Creating release notes. Creating the release. Creating snap-shots project logs. Distributing the release to the environments defined in the project's implementation roadmap.

Construction Pipeline Activities – Wrap up

Activity	Includes activities such as
Rework	Resolve issues found during system test or carried over from the prior stage. *Note: These activities will primary focus on features created during the prior stage*
Data Routines	Construction and testing the routines needed to load data in the new system. When upgrading an old system, it includes routines used to migrate, convert, and cleanse data from the old system to new. It follows the same construction and test activities defined in the prior table. *Notes: These activities will primary focus on data elements introduced during the prior stage*
Help Text & Support Documentation	Creating text for the help system. Creating user documentation Creating support documentation Formal and informal reviews, inspections, and desk checks Immediate rework of any identified issues *Note: These activities will primary focus on features created during the prior stage*

Recurring Activities

Activity	Includes activities such as
Project Management	Planning activities including estimation and scheduling. Creation and update of planning artifacts Planning and status meetings. Any work related to formal sponsor management, including formal change control activities Project Management by walking around (MBWA)

Activity	Includes activities such as
Technical Management	Activities performed while wearing a technical lead hat or technical lead activities if they are general in nature. Technical coaching Executing daily and release builds and automated testing Setting up formal reviews Dealing with issues and defects
Infrastructure / Miscellaneous	Creation and support of general team / development infrastructure Learning / Investigation This category can be used to cover any work that is not part of another work package. If significant amounts of time are placed here, model what is occurring and break into separate work packages
Staff Development	Creating and conduction training material to support deploying new technology and project practices. Delivering training to the development teams on new technology and project practices. Attending training.
General Quality Control	Any testing that spans multiple work packages (e.g., system testing or integration testing) Any testing that covers closed work packages (e.g., smoke, system, integration, or regression testing) Any reviews that span multiple work packages or cover closed work packages
General Rework	Investigating and fixing defects or issues in work packages that have already been closed Any other work that requires revisiting closed work packages to address problems or small changes.
Stage Reviews	Preparing for and conducting reviews at the end of each stage.

Status Report Template

The paragraphs written in the "Comment" style are for the benefit of the person writing the document and should be removed before the document is finalized.

The executive status report is an efficient mechanism for disseminating the state of a project.

*This template provides a starting point for creating a **one page** executive summary for communicating regular project status (weekly is recommended). This report is focuses on non-technical business questions.*

The areas of the template are intended to prompt specific information, but each project will want to customize the exact information to best meet its needs and management processes. When possible it is always preferable to provide quantitative measures of the project's performance. Qualitative descriptions can also be useful to support quantitative data or when such data is not available, but should be made as objective as possible.

The top of the report is a set of key project health questions. Other than the first line, status, the questions are all worded such that their answer will be 'No' if the project is on track.

The next table contains summary details on the state of the project. What data is reported, and how it is reported (e.g., absolute values vs. percentages) should be driven the needs of the stakeholders who will be consuming the report.

All reporting of actuals vs. plan should be against the latest project baseline

Executive Status Report

[Period Name, often Week X] – [Date Start] to [Date End] *(Report issued [Date])*

Key Questions

Status	What is the current project status?	Green
Schedule	Will target dates be missed?	No
Feature	Has the feature scope changed?	No
Effort	Has the planned effort changed?	No
Quality	Are there quality problems?	No
Risks	Have any new risks been identified?	No
Issues	Are there any issues requiring sponsor involvement?	No

Summary

Overview	A very brief summary of the state of the project. This summary should not simply repeat information found elsewhere in the report. It will often provide a qualitative perspective on quantitative information found elsewhere. If corrective action is being taken, it should be summarized here.
Tracking Information	Summarize key quantitative tracking information for the project. Focus on 3-4 key measures of project health such as earned value indexes, defect levels and rates, change and issue levels and rates, work packages status. etc.
Notable Events	Capture any significant events in the reporting period that are not discussed in the overview.

Consider displaying a small graphic of the latest milestone schedule or project lifecycle stages with completed and current milestones / phases indicated.

Top Project Risks

1. **Risk Name** – Brief description of risk and mitigation.
2. **Risk Name** – Brief description of risk and mitigation.
3. **Risk Name** – Brief description of risk and mitigation.

Sample Executive Status Report

Week 7 – July 30 to August 3, 2001 *(Report issued August 6, 2001)*

Key Questions

Status	What is the current project status?	Yellow
Schedule	Will target dates be missed?	No
Feature	Has the feature scope changed?	No
Effort	Has the effort changed?	No
Quality	Are there quality problems?	No
Risks	Have any new risks been identified?	No
Issues	Are there any issues requiring sponsor involvement?	No

Summary

Overview

The project is in yellow status because of problems to date finding resources; this should be resolved by next week.

Preparing for the M2 confirmation / M3 kickoff meeting which is scheduled for Nov 21st. The business transform team is on track to pass one record through the framework with in the next two weeks.

Tracking Information

Currently in the 5th week of milestone 2. On track to meet the M2 deadline.

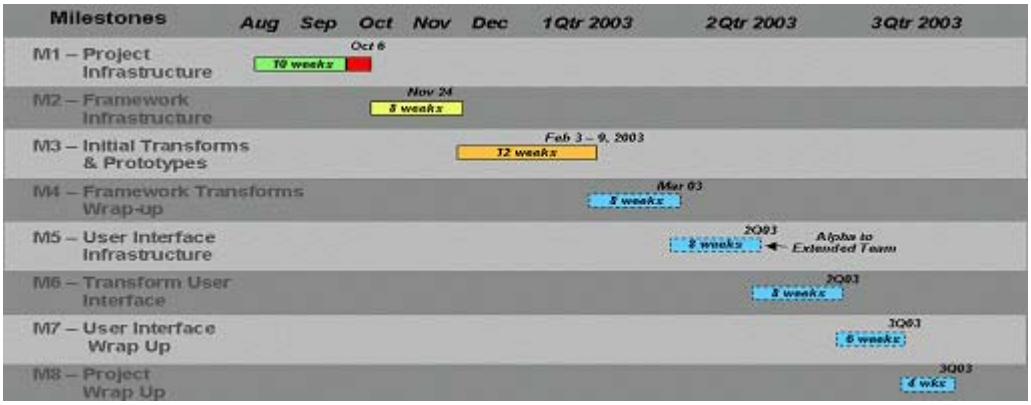
To date 137 work packages are closed, 45 are open, and 497 are remaining

The SPI or schedule performance index is at .98 (> 1.0 is good). The CPI or cost performance index (a.k.a. staff hours) is at 2.44 (> 1.0 is good)

Notable Events

Resource balancing plan was put into place. The only position we are short today is the Business Transform product tester.

Project management consultant visited last week to help with estimation.



Top Project Risks

4. **Staff Availability** – Appropriate staff are not available to fill all project positions. Working to reassign staff from other projects.
5. **Delayed Hardware Delivery** – Delayed delivery of hardware for server deployment will delay system coming online. Searching for temporary backup hardware.

PROJECT LOG TEMPLATE

The paragraphs written in the "Comment" style are for the benefit of the person writing the document and should be removed before the document is finalized.

Describes specific data about the project. Project logs are generally done at the end of each phase of a project.

Overview

Briefly summary the current phase and status of the project.

Major Decisions

Describe the major decisions made on the project, including date, background, and results.

Project Changes

Describe the changes to the schedule and effort that have been approved by the Change Control Board. Describe the number of change requests proposed and approved.

Approved Changes

Change Control Number	Description of Request

Schedule and Effort

Describe the current schedule and effort estimates.

Budget vs. Cost to Date

Major Activity	Baseline Hours to Date	Actual Hours to Date	Baseline Cost to Date	Actual Cost to Date
Activity A				
Activity B				

Describe and explain any deviations from Baseline dates and costs

Major Deliverables

Describe planned verses actual for each of the major deliverables on the project. This includes items such as the SRS, First Vertical Slice, etc.

Planned vs Actual Deliverables

Deliverables	Original Baseline Target Date	Revised Baseline Target Date	Actual Date
Deliverable A			
Deliverable B			

Describe and explain any deviations from Baseline dates.

Reviews

Describe the reviews completed on the project.

Project Measures and Metrics

Describe project measures and metrics including staff effort expended, size, open issues, etc.

Measure or Metric	Results
Staff Effort Expended to Date	
Lines of Code	
Number of Open Defects	

Issue List Pattern

This pattern defines a reusable table format for capturing a list of issues.

Overview

Issues are items which occur on a project which are not part of project planning. CxOne has developed the idea of corrective activity management (CAM) to cover the management of issues, along with change requests, defects, and risks. See *CxStand_ProjectManagement* for more details.

In most cases, a database will be used to track issues. However, with small projects a document-based list may be sufficient. In general, CxOne recommends using a defect tracking or database tool configured according to *CxPattern_CamDatabase.*

How to Use the Pattern

This table may be used as is or with modifications when creating project artifacts. Refer to *CxPattern_CamDatabase* for a description of how to manage information on the different columns.

Pattern

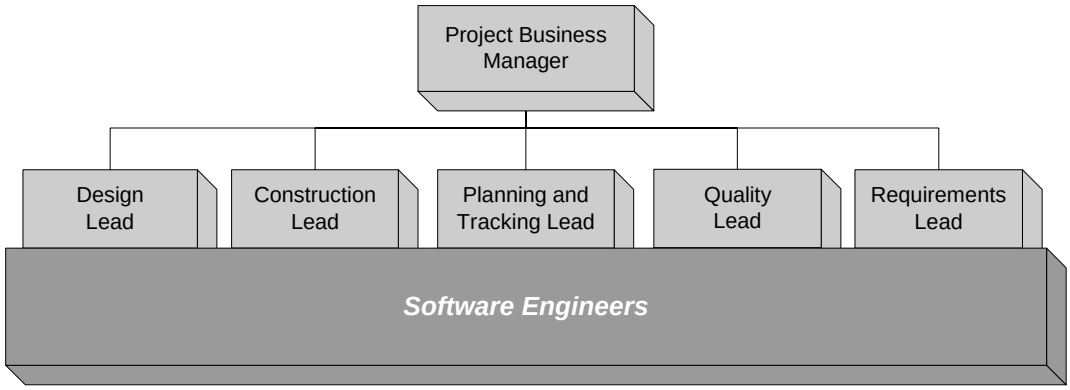
[TBD] Project Issue Tracking

ID	Description	Owner	State	Resolution

Roles & Responsibilities

Team of Leads

The diagram below describes CxOne positions for a typical software development project. These positions are commonly referred to as "hats". On smaller projects, engineers wear multiple hats and often will perform work as an engineer for other leads even if they are wearing a lead hat. Even on a two (or even one!) person project, calling out these responsibilities and using them to force thinking from different perspectives can be useful.



Project Team Structure

The responsibilities for planning and managing the work are split between a group of technical leads and the project business manager (PBM). Using an analogy with the movie industry, the PBM can be thought of as the producer (handles business issues), and the group of technical leads can be thought of as sharing the duties of a director (handles technical execution).

A pool of engineers carries out front-line work on the project. Depending on the project and organization, this pool may be a group of cross-trained people that share responsibilities, or specialists may be provided from different areas to work under specific leads.

Project Business Manager Checklist

CxOne Project Business Manager Checklist	
Project Management Tasks	Responsible for carrying out the project charter as defined by the project sponsor.
	Define overall project vision, scope, direction and constraints for technical leads.
	Create and maintain the project plan.
	Oversee creation of all other project artifacts.
	Drives risk management on the project.
	Make sure that all the work that needs to get done is actually getting done.
	Ensure work is progressing efficiently and effectively, i.e., the project is optimizing exe-cution of its goals based on its resources and constraints.
	Sets up revision control tools and processes for non-construction artifacts.
	Hold as needed status meetings with project sponsor and project stakeholders
Stakeholder Related Tasks	Hold weekly project status meeting with major stakeholders.
	Be a visible and accessible contact point for primary stakeholders.
	Resolve any disputes between stakeholders.
	Resolve any disputes between stakeholders and project team.
	Organize project travel for meetings as appropriate.
Team Related Tasks	Works with project sponsor to staff the project.
	Organize weekly project status meeting with team.
	Resolve disputes between technical leads.
	MBWA for general project issues.
	MBWA for general people issues
Planning and Tracking Tasks	Work with PTL to create weekly executive status reports.
	Work with PTL to create as needed detailed status reports.
	Ensure that scheduled work and prioritization of tasks meets all stakeholders needs.
CCB Tasks	Participation in the regular CCB meetings as needed.
	Ultimately responsible for resolving all CCB disputes.
Requirements Tasks	Ultimately responsible for resolving all requirements disputes
Release Creation Tasks	Make sure the release is released.
	Verify that appropriate stakeholders have received and are satisfied with a release

Planning and Tracking Lead Checklist

CxOne Planning and Tracking Lead Checklist	
Project Management Tasks	Coordinate the identification, estimation, prioritization and execution of work on the project.
	Help maintain the project risks list.
	Set up and maintain tools and processes for issue management.
	Inform the project business manager of risks to planned work and deviations from plan.
	Do MBWA and other management for planning and tracking issues.
Planning and Tracking Tasks	Be the primary technical resource for work breakdown, estimation, scheduling, and tracking is-sues.
	Assist in creation and maintenance of the project plan.
	Organizes all work breakdown and estimation for the project, coordinating input from other leads and team members.
	Assist in creation of the status reports.
	Create and maintain the work breakdown structure and project level estimates.
	Create and maintain the global business schedule.
	Create and maintain milestone and/or release cycle schedules.
	Organize weekly planning and tracking meeting with team.
	Review individual planning and tracking reports as appropriate.
	Oversee planning and implementation of planning and tracking tools.
CCB Tasks	Participate in the regular CCB meetings as needed.
	Coordinate estimates for CCB SCRs.
Release Creation Tasks	Monitor work being planned for a release to ensure that it will fit.
	Determine what planned work for a release was accomplished.
	Collect and record measures for project log and archive artifacts as appropriate.

Design Lead Checklist

CxOne Design Lead Checklist	
Project Management Tasks	Identify design risks. Do MBWA and other management for design issues.
Team Related Tasks	Provide guidance to team members
Planning and Tracking Tasks	Provides work breakdown for design related tasks. Assist in estimation activities as needed
CCB Tasks	Participate in the regular CCB meetings as needed.
Design Tasks	Be the primary technical resource for design issues related to the project. Create and maintain the architecture specification. Lead high level design activities. Ensure creation and maintenance of all design artifacts. Oversee and assist with low level design activities. Work with the requirements lead to ensure the requirements are sufficiently complete to sup-port design. Work with the construction lead to ensure feasibility of designs. Attempt to participate in all design discussions so there is one person who has the entire system design in their head (as much as that is possible).
Release Creation Tasks	Ensure design documents are reasonably up to date with any changes introduced in release, or SCR entered for future design document update. Collect and record measures for project log and archive artifacts as appropriate.

Construction Lead Checklist

CxOne Construction Lead Checklist	
Project Management Tasks	Identify technology and construction feasibility risks. Do MBWA and other management for construction issues.
Planning and Tracking Tasks	Assist in estimation activities as needed. Look for gold-plating and off-the-books work.
CCB Tasks	Participation in the regular CCB meetings as needed.
Construction Tasks	Be the primary technical resource for construction and technology issues on the project. Organize and drive project integration. Occasionally review source code organization to ensure integrity. Occasionally spot check source code content to ensure consistency and integrity. Keep construction documents and standards up to date, and ensure that processes and standards are being followed. Oversee construction tools and development environment.
Build Tasks	Design and oversee creation and maintenance of the build infrastructure. Sets up revision control tools and processes for construction artifacts. Integrate build with any deployment and test automation used on the project. If daily build or related automated tests and/or smoke tests fail, manage any issues to resolution.
Test Management	Oversee design, construction, and integration of unit and component level test code, both man-ual and automated
Release Creation Tasks	Coordinate marking and building of milestone builds. Provide release notes. Collect and record measures for project log and archive artifacts as appropriate.

Quality Lead Checklist

CxOne Quality Lead Checklist	
Project Management Tasks	
Stakeholder Related Tasks	General Quality Tasks
	Creates and maintains the quality and test plans.
	Work with other technical leads and project business manager to ensure optimal level and bal-ance of quality activities to achieve project goals.
	Be the primary technical resource on the project's quality activities.
	Do MBWA on quality issues.
	Quality Control Tasks
	Reviews
	Ensure reviews and review processes are meeting project quality goals.
	Organize and ensure successful execution of planned reviews.
	Test
	Ensure testing and test processes are meeting project quality goals.
	Setup and oversee automated testing infrastructure.
	Review and oversee all automated test creation, from system tests to unit tests.
	Ensure that TCSs are being created.
	Oversee the smoke test planning and execution.
	Oversee manual testing.
	Quality Assurance Tasks
	Define defect prevention processes including data collection and analysis.
	Ensure quality related data is collected.
	Oversee any other defect prevention activities.
	Project Management Tasks
	Identify quality related risks.
	Set up and maintain tools and processes for defect management.
	Participate in decisions to call work packages closed or tasks complete.
	Participation in CCB and other configuration management activities as needed.
	Work with the requirements lead to differentiate defects from change requests.
	Release Creation Tasks

CxOne Quality Lead Checklist

Team Related Tasks	
Planning and Tracking Tasks	Work with PTL to develop work breakdown for quality and testing activities. Work with PTL and team to estimate quality related tasks.
CCB Tasks	Participation in CCB and other configuration management activities as needed. Work with the requirements lead to differentiate defects from change requests.
Review Tasks	Ensure reviews and review processes are meeting project quality goals. Organize and ensure successful execution of planned reviews.
Test Tasks	Ensure testing and test processes are meeting project quality goals. Setup and oversee automated testing infrastructure. Review and oversee all automated test creation, from system tests to unit tests. Ensure that TCSs are being created. Oversee the smoke test planning and execution. Oversee manual testing.
Quality Assurance Tasks	Define defect prevention processes including data collection and analysis. Ensure quality related data is collected. Oversee any other defect prevention activities.
Release Creation Tasks	Ensure that system, release, and/or acceptance testing are complete. Characterize and/or certify a release, as per other project constraints. Collect and record measures for project log and archive artifacts as appropriate.

Requirements Lead Checklist

CxOne Requirements Lead Checklist	
Project Management Tasks	Identify requirements related risks.
	Set up and maintain tools and processes for change management.
	Do management by walking around and other management of requirements issues.
Planning and Tracking Tasks	Provides work breakdown for requirements related tasks.
	Estimate requirements related tasks.
CCB Tasks	Organize the regular CCB meetings.
	Prepare the issues list for CCB meetings.
	Create and maintain reports of the CCB meetings.
	Work with the QA lead to differentiate between defects and change requests.
	Escalate issues as necessary to the project business manager.
Release Creation Tasks	Note the requirements delivered for a release.
	Collect and record measures for project log and archive artifacts as appropriate.
Requirements Tasks	Be the primary resource on the project's requirements process.
	Works closely with stakeholders to discover, document, and maintain a set of requirements.
	Oversees any workshops, requirements prototyping, user interviews, modeling, or other requirements elicitation and analysis activities.
	Works closely with design and construction lead to determine feasibility of requirements.
	Creates and maintains the SRS, diagrams, database, and any other artifacts that capture system requirements.
	Perform requirements tracing.
	Oversee planning and implementation of requirements tools.

Sponsor Checklist

CxOne Sponsor Checklist	
Organizational Tasks	Define overall vision, direction, scope and constraints for the project business manager. Ensure project charter is created and accepted by all stakeholders. Ensure project charter is maintained and updated throughout the life of the project. Manage organizational risks of the project.
Project Oversight	Review status reports. Review project plans and deliverables, as appropriate, on a regular basis. Meet regularly with project business manager to discuss project.
External Stakeholders	Help resolve any disputes between internal project team and external stakeholders.
Team Related Tasks	Allocate staffing to project as appropriate. Assist resolving any project disputes as appropriate. MBWA for general people issues.
CCB Tasks	Approve changes in scope or budget

Sponsor Checklist

CxOne Sponsor Checklist	
Organizational Tasks	If appropriate, verify project charter with appropriate software engineering oversight authority.
Project Oversight	Review the planning and use of software engineering processes on the project. Review project risks, risk mitigation plans, and risk management practices. Review project status reports. Review project plans on a regular basis. As appropriate, review a cross section of project deliverables on a regular basis. Meet regularly with project business manager to discuss project. Work with project team to resolve any deviations from required organizational practices. Report any deviations from organizational practices or standards if not addressed by project team.
Project Tasks	Suggest software engineering processes and techniques to be used in project. Act as consultant, coach, and/or trainer on software engineering issues as appropriate. Organize and participate in project planning reviews, checkpoints, and retrospectives. Facilitate discussions and communication related to implementation of software engineering processes.
Auditing Tasks	If performing formal assessment or audit, prepare and deliver report as appropriate.

Subject Matter Experts

Possible SME Activities	
Planning	Refine the scope and limitations of the product.
	Define interfaces to other systems.
	Evaluate impact of new system on business operations.
	Define a transition path from different applications.
Requirements	Collect requirements from other users.
	Developer usage scenarios and use cases.
	Resolve conflicts between proposed requirements.
	Define implementation priorities.
	Specify quality and performance requirements.
	Evaluate user interface proposals.
Validation and Verification	Inspect requirements documents.
	Define user acceptance criteria.
	Develop test cases from usage scenarios.
	Provide test data sets.
	Perform Beta Testing.
User Aids	Write portions of the user manuals and help text.
	Prepare training materials for tutorials.
	Present product demonstrations to peers.
Change Management	Evaluate and prioritize defect corrections.
	Evaluate and prioritize enhancement requests.
	Evaluate the impact of proposed requirements changes on users and business processes.
	Participate in making change decisions.

CONTINUING PROFESSIONAL DEVELOPMENT

Construx's Professional Development Ladder

The purpose of Construx's professional ladder is to ensure that Construx is a world-class software engineering organization. It promotes professional development by individual engineers and focuses on improved organizational engineering knowledge and skills. The ladder embeds Construx's corporate values into our promotion and reward structure such that the road to promotion is clearly visible and aligned with self-improvement.

The ladder also provides a clear picture of organizational skill depth and supports equality in compensation among current and future employees. All technical employees have a ladder level and a professional development plan that outlines their career progression.

The ladder is organized around the Construx Knowledge Areas (CKAs). Within each of the knowledge areas, we recognize four levels of capability as outlined below.

- **Introductory** (Apprentice) – The employee performs or is capable of performing basic work in an area, generally under supervision. The employee is taking effective steps to develop his or her knowledge and skills.
- **Competence** (Practitioner) – The technical employee performs effective, independent work in an area, serves as a role model for less expert employees, and occasionally coaches others.
- **Leadership** (Leader) – The employee performs exemplary work in an area. The employee regularly coaches employees and provides project-level and possibly company-wide leadership. The employee is recognized within Construx as a major resource in the knowledge area.
- **Mastery** (Master) – The employee performs reference work in an area and has deep experience across multiple projects. The employee has generally taught seminars or classes or has written papers or books that extend the body of knowledge. The employee provides industry-level leadership and is recognized outside Construx for expertise in the area.

Construx's Professional Development Ladder supports a variety of career development paths. The structure enables each person to guide his or her career path based on their interests but within a structured framework. The following sample professional development plan have been provided to illustrate this.

The sample plan combines both knowledge and experience requirements as Construx believes that competence and effectiveness depends on a combination of the two. We believe that leading-edge knowledge in an engineering discipline is not possible unless it's grounded in experience, and leading-edge experience is not possible unless it's fully appraised of state-of-the-art knowledge.

More information about the professional development ladder and additional sample plans can be found on our website at:

http://www.construx.com/professionaldev/organization/pdl/

Sample PDP Snapshot for a Senior Project Manager

Construx also believes that a senior engineer requires a breath of knowledge areas. At Construx, a level-12 engineer is required to be at the leadership capability level (both knowledge and experience) in at least three of the knowledge areas. The following table illustrates a typical profile for a senior project manager.

Professional Development Snapshot for a Senior Project Manager

CKA Knowledge / Experience → / *Capability Level ↓*	CM		Construction		Design		Engineering Management		Maintenance		Process		Quality		Requirements		Testing		Tools and Methods	
Mastery																				
Leadership																				
Competence																				
Introductory																				
	K	*E*	*K*	*E*	*K*	*E*	*K*	*E*	*K*	*E*	*K*	*E*	*K*	*E*	*K*	*E*	*K*	*E*	*K*	*E*

Sample Engineering Management Ladder Requirements

Introductory Level

Knowledge Area: Management		**Capability Level: Introduction**
Required Accomplishments	**Reading Level**	**Description**
PDP Reading	I	"Software Engineering Code of Ethics and Professionalism", ACM/IEEE-CS
PDP Reading	A	"They Write the Right Stuff", Charles Fishman
PDP Reading	I	201 Principles of Software Development, Alan Davis
PDP Reading	I	Software Engineering, Ian Sommerville – Part 1 + chapters 22 & 23
PDP Reading	I	Software Engineering: A Practioners Approach, Pressman – Chapter 4
PDP Reading	A	Software Project Survival Guide, Steve McConnell
Work Activity		Learn an estimation technique
Work Activity	I	Read CxOne materials for this CKA
Work Activity		Review and discuss project artifacts related to this area

Competence Level

Knowledge Area: Management		Capability Level: Competence
Required Accomplishments	**Reading Level**	**Description**
PDP Reading	I	SWEBOK, Chapter 8 - Engineering Management
PDP Reading	A	"No Silver Bullets--Essence and Accidents of Software Engineering", Fred Brooks
PDP Reading	A	"Programmer Performance and the Effects of the Workplace", DeMarco and Lister
PDP Reading	I	"Software's Chronic Crisis", Wayt Gibbs
PDP Reading	A	Manager's Handbook for Software Development, NASA Goddard Space Flight Center
PDP Reading	A	Rapid Development, Steve McConnell
Training		Attend Project Management Boot Camp
Training		Attend Software Estimation In Depth
Work Activity		Create a weekly status report
Work Activity		Lead an estimation activity
Work Activity		Participate as a reviewer for a project's artifacts in this area
Work Activity		Participate in the creation of a project charter
Work Activity		Participate in the creation of a project estimate
Work Activity		Participate in the creation of a project plan
Work Activity		Perform significant work in this area on a project
Work Activity		Proficient with individual bottom-up estimation techniques
Work Activity		Proficient with individual status reporting techniques
Work Activity		Proficient with work planning (as defined by CxOne, includes WBS, estimation, and EVM)
Work Activity	A	Read CxOne materials for this CKA

Leadership Level

Knowledge Area: Management		Capability Level: Leadership
Required Accomplishments	**Reading Level**	**Description**
Certification		IEEE CSDP
Certification		PMI Certification

Knowledge Area: Management		Capability Level: Leadership
Required Accomplishments	**Reading Level**	**Description**
PDP Reading	A	Project Management Body of Knowledge (PMBOK), PMI
Training		Attend How to be Agile Without Being Extreme
Training		Attend Software Measurement In-Depth
Training		Attend Risk Management In-Depth
Work Activity		Create a business (or milestone) schedule for a significant project
Work Activity		Create a detailed schedule for a milestone
Work Activity		Create a project charter with business case
Work Activity		Create a project plan for a large and a small project
Work Activity		Create a top-down project estimate at project inception
Work Activity		Create a work plan for a significant project
Work Activity		Lead issue management for a significant project
Work Activity		Lead planning, estimation, and tracking activities for a significant project
Work Activity		Participate in consulting / coaching work in this area
Work Activity		Perform significant work in this area on multiple diverse projects
Work Activity		Proficient in group estimation techniques (e.g., wide-band delphi)
Work Activity		Proficient with analogy estimation techniques
Work Activity		Proficient with critical-path, critical-chain, and rolling-wave scheduling techniques
Work Activity		Proficient with formal issue management techniques
Work Activity		Proficient with formal risk management techniques
Work Activity		Proficient with historical data collection techniques
Work Activity		Proficient with lifecycle selection, customization, and planning
Work Activity		Proficient with parametric estimation techniques
Work Activity		Proficient with project status reporting techniques

Knowledge Area: Management		Capability Level: Leadership
Optional Accomplishments	**Reading Level**	**Description**
PDP Reading	A	"A Method of Programming Measurement and Estimation", Walston and Felix
PDP Reading	A	"One More Time: How Do You Motivate Employees", Frederick Herzberg
PDP Reading	A	"Understanding and Controlling Software Costs", Barry Boehm and Phillip Papaccio
PDP Reading	I	A Managers Guide to Software Engineering, Roger Pressman
PDP Reading	I	Agile and Iterative Development: A Manager's Guide, Craig Larman
PDP Reading	I	Agile Project Management with Scrum, Ken Schwaber
PDP Reading	A	Assessment and Control of Software Risks, Capers Jones
PDP Reading	A	Balancing Agility and Discipline, Barry Boehm and Richard Turner
PDP Reading	A	Becoming a Technical Leader, Gerald Weinberg
PDP Reading	I	Building the Project Driven Enterprise: Using Lean Project Management, by Ronald Mascitelli
PDP Reading	A	Constantine on Peopleware, Larry Constantine
PDP Reading	A	Controlling Software Projects, Tom DeMarco
PDP Reading	A	Creating a Software Engineering Culture, Karl Wiegers
PDP Reading	A	Creating the High-Performance Team, Steve Buchholz and Tom Roth
PDP Reading	A	Dealing with People You Can't Stand, Rick Brinkman and Rick Kirschner
PDP Reading	A	Debugging the Development Process, Steve Maguire
PDP Reading	A	Dynamics of Software Development, Jim McCarthy
PDP Reading	I	Extreme Programming Explained (2nd Edition), Kent Beck
PDP Reading	I	Extreme Programming Refactored, Matt Stephens and Doug Rosenberg
PDP Reading	I	Fast Forward MBA in Project Management (2nd Edition), Eric Versuh
PDP Reading	A	Getting to Yes, Robert Fisher
PDP Reading	A	High Output Management, Andy Grove
PDP Reading	A	How to Get Your Point Across in 30 Seconds or Less, Milo O. Frank

Knowledge Area: Management		Capability Level: Leadership
Optional Accomplishments	**Reading Level**	**Description**
PDP Reading	I	How to Make Meetings Work, Michael Doyle
PDP Reading	A	IEEE project management standards
PDP Reading	I	IEEE software lifecycle standards
PDP Reading	A	Make Presentations with Confidence, Vivian Buchan
PDP Reading	A	Managing a Programming Project 3d Ed, Philip Metzger and John Boddie
PDP Reading	A	Measures for Excellence, Lawrence H. Putnam and Ware Myers
PDP Reading	A	Microsoft Secrets, Cusumano
PDP Reading	A	Mythical Man-Month 2nd Ed, Fred Brooks
PDP Reading	A	Peopleware, 2nd Ed, Tom DeMarco and Tim Lister
PDP Reading	A	Principle-Centered Leadership, Steven Covey
PDP Reading	A	Principles of Software Engineering Management, Tom Gilb
PDP Reading	I	Project Retrospectives, Norm Kerth
PDP Reading	I	Professional Software Development, Steve McConnell
PDP Reading	I	Recommended Approach to Software Development, NASA Goddard Space Flight Center
PDP Reading	I	Return on Software, Steve Tockey
PDP Reading	I	Quality Software Management Vol 3, Gerald Weinberg
PDP Reading	I	Quality Software Management Vol 4, Gerald Weinberg
PDP Reading	A	Software Engineering Economics 2ed Barry W. Boehm
PDP Reading	I	Software Engineering Project Management, Richard H. Thayer
PDP Reading	I	Software Management, Donald Reifer
PDP Reading	I	Software Risk Management, Barry Boehm
PDP Reading	A	Software Runaways, Robert Glass
PDP Reading	I	The Art of Project Management, Scot Berkun
PDP Reading	I	The Deadline, Tom DeMarco
PDP Reading	I	The Goal, Eliyahu Goldratt
PDP Reading	A	The Product Manager's Handbook, Linda Gorchels
PDP Reading	A	Wicked Problems Righteous Solutions, DeGrace and Hulet
Prof. Activity		Create evening, weekend, or college course in this area

Knowledge Area: Management		Capability Level: Leadership
Optional Accomplishments	**Reading Level**	**Description**
Prof. Activity		Participate in an industry committee, panel, group, standards board, etc.
Prof. Activity		Present at a conference
Prof. Activity		Publish an article in a minor publication
Prof. Activity		Publish an article in major or peer-reviewed publication
Prof. Activity		Review a book manuscript
Prof. Activity		Review articles for IEEE Software or similar publication
Prof. Activity		Teach an evening, weekend, or college class
Work Activity		Create a customized lifecycle for a significant project
Work Activity		Create a proposal for an outsourced project
Work Activity		Create new CxOne materials for this CKA
Work Activity		Create professional course in this area
Work Activity		Lead a staff performance review
Work Activity		Lead consulting / coaching work in this area
Work Activity		Lead formal risk management activities for a significant project
Work Activity		Lead the business management for a significant project
Work Activity		Mastery of a project management domain / technique / methodology / tool
Work Activity		Proficient with EVM techniques that are in addition to work planning
Work Activity		Proficient with software contractual knowledge and techniques
Work Activity		Proficient with statistical analysis and statistical process control techniques
Work Activity		Teach professional course in this area

Mastery Level

Knowledge Area: Management		Capability Level: Mastery
Required Accomplishments	**Reading Level**	**Description**
Prof. Activity		Create significant advancements to the art and science of this area
Work Activity		Create significant advancements to the art and science of this area

Knowledge Area: Management		Capability Level: Mastery
Optional Accomplishments	**Reading Level**	**Description**
Prof. Activity		Publish a respected software engineering book

.

Take Home Points

Take Home Points (cont)

Back to Work Action Plan

What Do You Plan to Do Within the **Next Week**?

1. _____

2. _____

3. _____

4. _____

5. _____

What Do You Plan to Do Within the **Next Month**?

1. _____

2. _____

3. _____

4. _____

5. _____

What Do You Plan to Do Within the **Next Two Quarters**?

1. _____

2. _____

3. _____

4. _____

5. _____

Where Do You Want to Be in the **Next Year**?

1. _____

2. _____

3. _____

4. _____

5. _____

Contact Information

For more information about the services we offer,

please email us at sales@construx.com,

visit our Internet site at www.construx.com,

call us at 425.636.0100,

fax us at 425.636.0159,

or send post to

Construx Software

10900 NE 8th Street, Suite 1350

Bellevue, WA 98004

Construx
SOFTWARE