The Rational Unified Process Phase and milestone Concept For A Project

The software lifecycle concept of the Rational Unified Process is decomposed over time into four sequential phases, each concluded by a major milestone; each phase is essentially a span of time between two major milestones, shown in Figure 1. At each phase-end an assessment is performed (called an Activity: Lifecycle Milestone Review, given in Appendix A) to determine whether the objectives of the phase have been met. A satisfactory assessment allows the project to move to the next phase.

Inception |	**Elaboration** |	**Construction** |	**Transition** |
^	^	^	^
Lifecycle objectives milestones	Lifecycle architecture milestones	Initial operational capability milestone	Product release milestone

____________________________Time____>

Figure 1: The Phase and Milestones of a Project

Planning Phases

All phases are not identical in terms of schedule and effort. Although this varies considerably depending on the project, a typical initial development cycle for a medium-sized project should anticipate the following distribution between effort and schedule:

	Inception	Elaboration	Construction	Transition
Effort	~5 %	20 %	65 %	10%
Schedule	10 %	30 %	50 %	10%

Figure 2: Phase Effort and Schedule

which can also be depicted graphically as:

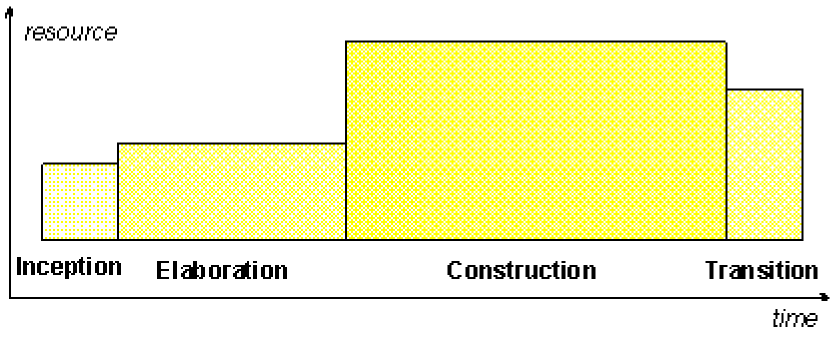


Figure 3 Plot of Phase Effort and Schedule

For an evolution cycle, the inception and elaboration phases would be considerably smaller. Tools, which can automate some portion of the Construction effort, can mitigate this, making the construction phase much smaller than the inception and elaboration phases together.

One pass through the four phases is a **development cycle**; each pass through the four phases produces a **generation** of the software. Unless the product "dies," it will evolve into its next generation by repeating the same sequence of inception, elaboration, construction and transition phases, but this time with a different emphasis on the various phases. These subsequent cycles are called **evolution cycles.** As the product goes through several cycles, new generations are produced.

Evolution cycles may be triggered by user-suggested enhancements, changes in the user context, changes in the underlying technology, reaction to the competition, and so on. Evolution cycles typically have much shorter Inception and Elaboration phases, since the basic product definition and architecture are determined by prior development cycles. Exceptions to this rule are evolution cycles in which a significant product or architectural redefinition occurs.

Phase 1: Inception

Objectives

The overriding goal of the inception phase is to achieve concurrence among all stakeholders on the lifecycle objectives for the project. The inception phase is of significance primarily for new development efforts, in which there are significant business and requirements risks, which must be addressed before the project can proceed. For projects focused on enhancements to an existing system, the inception phase is more brief, but is still focused on ensuring that the project is both worth doing and possible to do.

The primary objectives of the inception phase include:

- Establishing the project's software scope and boundary conditions, including an operational vision, acceptance criteria and what is intended to be in the product and what is not.
- Discriminating the critical use cases of the system, the primary scenarios of operation that will drive the major design trade-off.
- Exhibiting, and maybe demonstrating, at least one candidate architecture against some of the primary scenarios
- Estimating the overall cost and schedule for the entire project (and more detailed estimates for the elaboration phase that will immediately follow)
- Estimating potential risks (the sources of unpredictability)
- Preparing the supporting environment for the project.

Essential activities

- **Formulating the scope of the project**. This involves capturing the context and the most important requirements and constraints to such an extent that you can derive acceptance criteria for the end product.

- **Planning and preparing a business case**. Evaluating alternatives for risk management, staffing, project plan, and cost/schedule/profitability trade-off.
- **Synthesizing a candidate architecture**, evaluating trade-off in design, and in make/buy/reuse, so that cost, schedule and resources can be estimated. The aim here is to demonstrate feasibility through some kind of proof of concept. This may take the form of a model which simulates what is required, or an initial prototype which explores what are considered to be the areas of high risk. The prototyping effort during inception should be limited to gaining confidence that a solution is possible - the solution is realized during elaboration and construction.
- **Preparing the environment for the project**, assessing the project and the organization, selecting tools, and deciding which parts of the process to improve.

Milestone: Lifecycle Objectives

At the end of the inception phase is the first major project milestone or **Lifecycle Objectives Milestone**. At this point, you examine the lifecycle objectives of the project, and decide either to proceed with the project or to cancel it.

Evaluation Criteria

- Stakeholder concurrence on scope definition and cost/schedule estimates
- Agreement that the right set of requirements have been captured and that there is a shared understanding of these requirements.
- Agreement that the cost/schedule estimates, priorities, risks, and development process is appropriate.
- All risks have been identified and a mitigation strategy exists for each.

The project may be aborted or considerably re-thought if it fails to reach this milestone.

Artifacts

Essential Artifacts (in order of importance)	**State at milestone**
Vision	The core project's requirements, key features, and main constraints are documented.
Business Case	Defined and approved.
Risk List	Initial project risks identified.
Software Development Plan	Initial phases, their duration's and objectives identified. Resource estimates (specifically the time, staff, and development environment costs in particular) in the Software Development Plan must be consistent with the Business Case. The resource estimate may encompass either the entire project through delivery, or only an estimate of resources needed to go through the elaboration phase. Estimates of the resources required for the entire project should be viewed as very rough, a "guesstimate" at this point. This estimate is updated in each phase and each iteration, and becomes more accurate with each iteration. Depending on the needs of the project, one or more of the enclosed "Plan" artifacts may be conditionally completed. In addition, the enclosed "Guidelines" artifacts are typically in at least a "draft" form.
Iteration Plan	Iteration plan for first Elaboration iteration completed and reviewed.
Product Acceptance Plan	Reviewed and baselined; will be refined in subsequent iterations as additional requirements are discovered.
Development Case	Adaptations and extensions to the Rational Unified Process, documented and reviewed.
Project-Specific Templates	The document templates used to develop the document artifacts.
Use-Case Modeling Guidelines	Baselined.
Tools	All tools to support the project are selected. The tools necessary for work in Inception are installed.
Glossary	Important terms defined; glossary reviewed.
Use-Case Model (Actors, Use Cases)	Important actors and use cases identified and flows of events outlined for only the most critical use cases.
Optional Artifacts	**State at milestone**
Domain Model (a.k.a. Business Object Model)	The key concepts being used in the system, documented and reviewed. Used as an extension to the Glossary in cases where there are specific relationships between concepts that are essential to capture.
Prototypes	One or more proof of concept prototypes, to support the Vision and Business Case, and to address very specific risks.

Sample Iteration Plan: Inception Phase

This illustration shows how a project begins, and how the various workflows relate. It is constructed from the Workflow Details as they would appear at the time of the first iteration of the project. The intent is to indicate dependencies and show where workflows occur in parallel. The lengths of the bars in the chart (indicating duration) have no absolute significance. For example, it is not intended to convey that Conceive New Project and Plan Test must have the same duration. There is also no intention to suggest the application of a uniform level of effort across the duration of the workflows. An indication of the relative effort can be seen in the Process Overview found in Appendix B. Figure 4 gives the inception phase workflow detail.

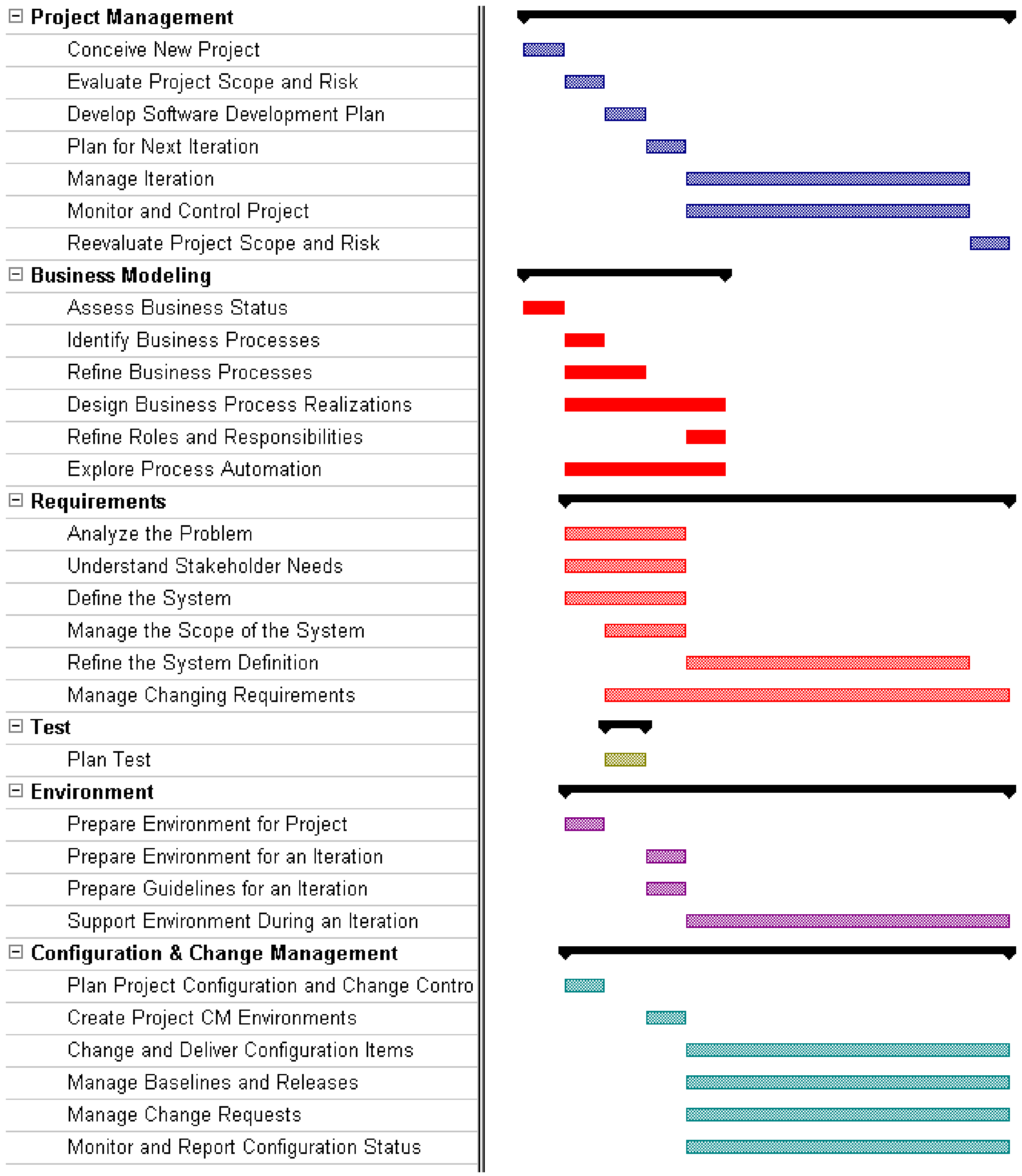


Figure 4 Inception Phase Workflow Detail

Sample Iteration Plan

Preliminary: Define the Business Context (optional)	In cases where the system is being built to support a new or significantly changed business process, some context-setting business engineering can help to better define the environment in which the system will operate. This is especially useful if the stakeholders are having difficulties expressing the requirements on the system needed to support the new or changed business process, or have difficulty separating what the new system will do as opposed to what the new business process will do. Defining the business context starts with Workflow Detail: Identify Business Processes. Prioritize those business processes that affects the system being built, and detail those according to Workflow Detail: Refine Business Process Definitions. Workflow Detail: Design Business Process Realizations and the Workflow Details: Refine Roles and Responsibilities shows how you further refine your understanding of the responsibilities that need to be carried out by the organization. In parallel with building the process realizations, you need to look at types of system sort, as described in Workflow Detail: Explore Process Automation. The degree of business engineering performed depends on the desired results. If the purpose of business engineering is merely to set context for the system, the effort should be restricted to the subset of the business which will be supported by the system to be developed. Further business engineering, while perhaps valuable for other reasons, tends to be de-focusing for the system development team.
Start up: Define the vision and scope of the system.	The Stakeholders of the system to be developed, working with System Analysts, define the vision and the scope of the project (see Workflow Detail: Analyze Problem in the Requirements workflow, and the Artifact: Vision). The driving factor to consider in this effort is the user's needs and expectations. Also considered are constraints on the project, such as platforms to be supported, and external interfaces. Based on the early sketches of the Vision, start to define the Artifact: Business Case and document the important risks in the Artifact: Risk List.
Outline and clarify the functionality that is to be provided by system.	Conduct sessions to collect stakeholders opinions on what the system should do. This can be done using various techniques (See Work Guidelines: Storyboarding and Work Guidelines: Brainstorming). You can also include building an initial outline of the Artifact: Use-Case Model in this session. The Artifact: Glossary will likely be started to simplify the maintenance of the use-case model, and to keep it consistent. See Workflow Detail: Understand Stakeholder Needs. The main result of these sessions is the Artifact: Stakeholder Requests and an outline of the Artifact: Use-Case Model.
Consider the feasibility of the project, and outline the project plan.	With the input from the use-case modeling, translate the Artifact: Vision into economic terms, updating the Artifact: Business Case, factoring in the project's investment costs, resource estimates, the environment needed, and success criteria (revenue projection and market recognition). Update the Artifact: Risk List to refer to the identified use cases and add new identified risks. Establish the initial Artifact Software Development Plan, mapping out the phases (Inception, Elaboration, Construction, and Transition), and major milestones.
Prepare the environment	Assess the current state of the project and its surrounding organization (see Workflow Detail: Prepare Environment for Project). The Worker: Process Engineer develops a first version of the Artifact: Development Case. The Worker: Tool Specialist selects tools for the project, and sets up the tools necessary to support the Requirements work. The Worker: System Analyst produces a first draft of the Artifact: Use-Case Modeling Guidelines.
Refine the project plan.	At this stage, the stakeholders of the system to be developed should have a fairly good understanding of its vision and the feasibility of the project. An order of priority among features and use cases is established (see Workflow Detail: Manage the Scope of the System, Artifact: Iteration Plan, and Artifact: Vision). The Worker: Project

	Manager refines the Artifact Software Development Plan, mapping out a set of iterations using the prioritized use cases and associated risks (see Artifact: Risk List). The plans developed at this point are refined after each subsequent iteration and become more accurate as iterations are completed. Note: this is a key differentiator in using this process - recognizing that initial project plan estimates are rough estimates, but that those estimates become more realistic as the project progresses and there are real metrics on which to base estimates; successive refinement of the project and iterations plans is both expected and essential.

Result

The result of this initial iteration is a first cut at the Artifact: Vision and the Artifact: Business Case of the project, as well as the scope of the project and the Artifact Software Development Plan. The stakeholders initiating the project should have a good understanding of the project's return on investment (ROI), that is, what is returned at what investment costs. Given this knowledge a go/no go decision can be taken.

Subsequent Iterations In The Inception Phase

In cases where the project involves new product rollout or creation of new technology, subsequent iterations may be needed to further define the scope of the project, the risks and the benefits. This may involve further enhancing the use-case model, business case, risk list or project and iteration plans. Extension of the Inception phase may also be advisable in cases where both the risk and the investment required are high, or where the problem domain is new or the team inexperienced.

Phase 2: Elaboration

Objectives

The goal of the elaboration phase is to baseline the architecture of the system to provide a stable basis for the bulk of the design and implementation effort in the construction phase. The architecture evolves out of a consideration of the most significant requirements (those that have a great impact on the architecture of the system) and an assessment of risk. The stability of the architecture is evaluated through one or more architectural prototypes.

The primary objectives of the elaboration phase include:

- To ensure that the architecture, requirements and plans are stable enough, and the risks sufficiently mitigated to be able to predictably determine the cost and schedule for the completion of the development. For most projects, passing this milestone also corresponds to the transition from a light-and-fast, low-risk operation to a high cost, high-risk operation with substantial organizational inertia.
- To address all architecturally significant risks of the project
- To establish a baselined architecture derived from addressing the architecturally significant scenarios, which typically expose the top technical risks of the project.
- To produce an evolutionary prototype of production-quality components, as well as possibly one or more exploratory, throw-away prototypes to mitigate specific risks such as:

    - design/requirements trade-off
    - component reuse
    - product feasibility or demonstrations to investors, customers, and end-users.

- To demonstrate that the baselined architecture will support the requirements of the system at a reasonable cost and in a reasonable time.
- To establish a supporting environment.

In order to achieve this primary objectives, it is equally important to set up the supporting environment for the project. This includes creating a development case, create templates, guidelines, and setting up tools.

Essential activities

- **Defining, Validating and Baselining the architecture** as rapidly as practical.
- **Refining the Vision**, based on new information obtained during the phase, establishing a solid understanding of the most critical use cases that drive the architectural and planning decisions.
- **Creating and Baselining detailed iteration plans for the construction phase**.
- **Refining the development case and putting in place the development environment**, including the process, tools and automation support required to support the construction team.
- **Refining the architecture and selecting components**. Potential components are evaluated and the make/buy/reuse decisions sufficiently understood to determine the construction phase cost and schedule with confidence. The selected architectural components are integrated and assessed against the primary scenarios. Lessons learned from these activities may well result in a redesign of the architecture, taking into consideration alternative designs or reconsideration of the requirements.

Milestone: Lifecycle Architecture

At the end of the elaboration phase is the second important project milestone, the **Lifecycle Architecture Milestone**. At this point, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks.

Evaluation Criteria

- The product Vision and requirements are stable.
- The architecture is stable.
- Executable prototypes have demonstrated that the major risk elements have been addressed and have been credibly resolved.
- The iteration plans for the construction phase are of sufficient detail and fidelity to allow the work to proceed.
- The iteration plans for the construction phase are supported by credible estimates.
- All stakeholders agree that the current vision can be met if the current plan is executed to develop the complete system, in the context of the current architecture.
- Actual resource expenditure versus planned expenditure are acceptable.

The project may be aborted or considerably re-thought if it fails to reach this milestone.

Artifacts

Essential Artifacts (in order of importance)	State at milestone
Prototypes	One or more executable architectural prototypes have been created to explore critical functionality and architecturally significant scenarios. See the note below on the role of prototyping.
Risk List	Updated and reviewed. New risks are likely to be architectural in nature, primarily relating to the handling of non-functional requirements.
Development Case	Refined based on early project experience. The development environment, including the process, tools and automation support required to support the construction team will have been put in place.
Project-Specific Templates	The document templates used to develop the document artifacts.
Tools	The tools used to support the work in Elaboration are installed.
Software Architecture Document	Created and baselined, including detailed descriptions for the architecturally significant use cases (use-case view), identification of key mechanisms and design elements (logical view), plus definition of the process view and the deployment view (of the Deployment Model) if the system is distributed or must deal with concurrency issues.
Design Model (and all constituent artifacts)	Defined and baselined. Use-case realizations for architecturally significant scenarios have been defined and required behavior has been allocated to appropriate design elements. Components have been identified and the make/buy/reuse decisions sufficiently understood to determine the construction phase cost and schedule with confidence. The selected architectural components are integrated and assessed against the primary scenarios. Lessons learned from these activities may well result in a redesign of the architecture, taking into consideration alternative designs or reconsideration of the requirements.
Data Model	Defined and baselined. Major data model elements (e.g. important entities, relationships, tables) defined and reviewed.
Implementation Model (and all constituent artifacts, including Components)	Initial structure created and major components identified and prototyped.
Vision	Refined, based on new information obtained during the phase, establishing a solid understanding of the most critical use cases that drive the architectural and planning decisions.
Software Development Plan	Updated and expanded to cover the Construction and Transition phases.
Guidelines, such as Design Guidelines and Programming Guidelines.	The guidelines used to support the work.
Iteration Plan	Iteration plan for the construction phase completed and reviewed.
Use-Case Model (Actors, Use Cases)	A use-case model (approximately 80% complete) - all use cases having been identified in the use-case model survey, all actors having been identified, and **most** use-case descriptions (requirements capture) having been developed.
Supplementary Specifications	Supplementary requirements capturing the non functional requirements are documented and reviewed.

Optional Artifacts	**State at milestone**
Business Case	Updated if architectural investigations uncover issues that change fundamental project assumptions.
Analysis Model	May be developed as a formal artifact; frequently not formally maintained, evolving into an early version of the Design Model instead.
Training Materials	User Manuals & other training materials. Preliminary draft, based on use cases. May be needed if the system has a strong user interface aspect.

The Role of Prototyping

The Rational Unified Process gives the architect and project manager the freedom to construct prototypes of several types (see Concepts: Prototypes) as a risk reduction strategy. Some of these prototypes may be purely exploratory, and are subsequently discarded. However, it is likely (certainly for larger or unprecedented systems) that the architecture will have been constructed as a series of evolutionary prototypes - covering different issues as elaboration proceeds - and by the end of elaboration, will have culminated in an integrated, stable architectural base. We do not mean to suggest here that the prototyping effort during elaboration should result in a set of architectural fragments, which need not be integrated.

Sample Iteration Plan: Elaboration Phase

This illustration shows the relationship of the workflows in an early elaboration iteration. It is constructed from the Workflow Details, as they would appear at that time. The intent is to indicate dependencies and show where workflows occur in parallel. The lengths of the bars in the chart (indicating duration) have no absolute significance. For example, it is not intended to convey that Plan for Next Iteration and Manage the Scope of the System must have the same duration. There is also no intention to suggest the application of a uniform level of effort across the duration of the workflows. An indication of the relative effort can be seen in the Process Overview found in Appendix B. Figure 5 gives the elaboration phase workflow detail.

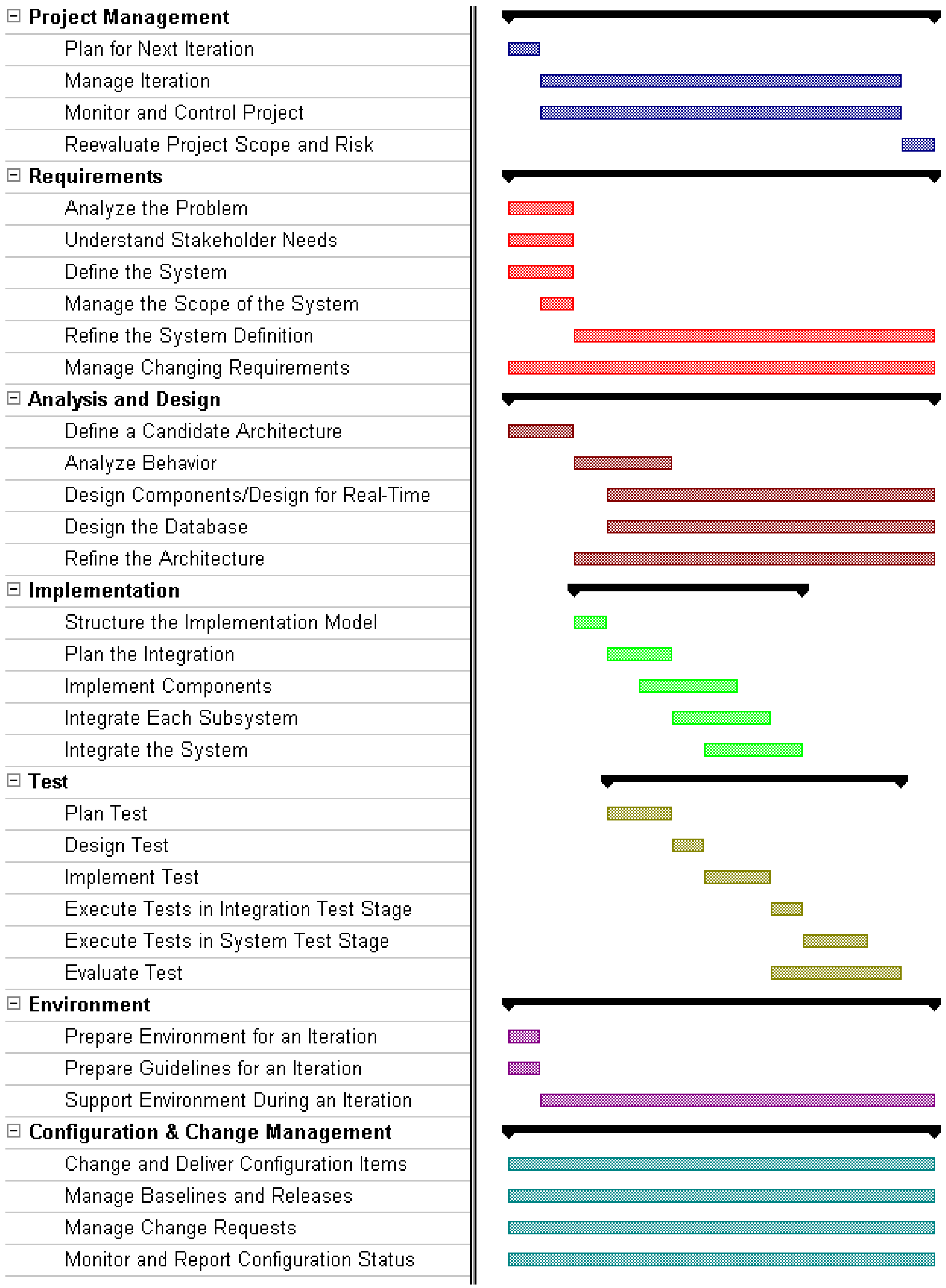


Figure 5: Elaboration Phase Workflow Detail

Note that although this is a plan for a single iteration, not all Requirements and Analysis and Design work performed during this iteration is intended for Implementation and Test in this iteration. This explains why the relative effort, within an iteration, for Requirements, Analysis and Design, Implementation and Test, changes through the life-cycle. However, the Iteration Plan will dictate what requirements are explored and refined and what components are designed, even if they are intended for Implementation and Test in a later iteration.

At the start of the elaboration phase, Inception Phase has been completed and the project has been funded. An initial Artifact: Software Development Plan exists, along with preliminary Artifact: Iteration Plans for at least the Elaboration Phase. The requirements of the system, captured by the Artifact: Use-Case Model and Artifact: Supplementary Specifications, have been briefly outlined.

Sample Iteration Plan

Start up: Outline the iteration plan, risks, and architectural objectives.	The Worker: Project Manager starts by sketching a an Artifact: Iteration Plan for the current iteration, based on the outlined Artifact: Software Development Plan during the Initial Iteration in Inception. Evaluation criteria for the architecture are outlined by the Worker: Architect in the Artifact: Software Architecture Document, taking into consideration the "architectural risks" that are to be mitigated (see Artifact: Risk List). Remember that one of the goals of Elaboration is establishing a robust, executable architecture; the plan for doing this needs to be developed in the initial Elaboration iteration.
Environment: Prepare environment for the iteration	The Worker: Process Engineer and the Worker: Tool Specialist prepare the environment for the iteration (see the Workflow Detail: Prepare Environment for an Iteration). An important input is the evaluation of the previous iteration. The Worker: Process Engineer completes the Artifact: Development Case and tailor templates (see Artifact: Project-Specific Templates), to be ready for the iteration, by tailoring (at least) the Analysis & Design workflow and the Implementation workflow. The Worker: Tool Specialist sets up the tools (see Artifact: Tools) to be used in the iteration. If necessary, produce Artifact: Tools Guidelines. The relevant guidelines are developed (see Workflow Detail: Prepare Guidelines for an Iteration).
Requirements: Decide what will "drive" the development of the architecture.	The Worker: Architect and the Worker: Project Manager then determine which use cases and/or scenarios should be addressed in the current iteration; these use cases and/or scenarios drive the development of the architecture (see Workflow Detail: Manage Scope of the System in the Requirements Workflow). The Artifact: Iteration Plan created in previous step should be updated accordingly.
Understand the "drivers" in detail, if necessary; inspect results.	A number of Worker: Use-Case Specifiers then describe in detail the architecturally significant subsets of the selected use cases/scenarios (see Workflow Detail: Refine the System Definition in the Requirements workflow). As the model evolves, the Worker: System Analyst may restructure the Artifact: Use-Case Model to improve the comprehensibility of the model. The changes to the Artifact: Use-Case Model are then reviewed and approved (see Workflow Detail: Manage Changing Requirements in the Requirements Workflow)

The "drivers" of the architecture are reconsidered according to new information; risks also need to be reconsidered.	The use-case view is revisited again by the Worker: Architect, taking into consideration new use-case descriptions, and possibly a new structure of the Artifact: Use-Case Model (revisit Workflow Detail: Manage Scope of the System in the Requirements Workflow). The task now is to select what set of use cases and/or scenarios should be analyzed, designed and implemented in the current iteration. Note again that the development of these use cases and/or scenarios set the software architecture. The Worker: Project Manager again updates the current iteration plan accordingly (see Artifact: Iteration Plan), and might also reconsider risk management, because new risks might have been made visible according to new information (see Artifact: Risk List).
Use-Case Analysis: Find obvious classes, do an initial (high-level) subsystem partitioning, and start looking at the "drivers" in detail.	To get a general feeling of the obvious classes needed, the Worker: Architect then considers the system requirements, the glossary, the use-case view (but not use case descriptions), and the team's general domain knowledge to sketch the outline of the subsystems, possibly in a layered fashion (see Activity: Identify Design Elements in the Analysis & Design Workflow). The analysis mechanisms (common solutions to frequent analysis problems) are also identified by the architect. In parallel with this effort, a team of Worker: Designers, possibly together with the architect, will start finding Artifact: Analysis Classes for this iteration's use cases and/or scenarios, as well as beginning to allocate responsibilities to the identified classes and analysis mechanisms. The designers will use the obvious classes found by the architect as input. Then, a number of designers refine the classes identified in the previous step by allocating responsibilities to the classes, and updating their relationships and attributes. It is determined in detail how the available analysis mechanisms are used by each class. When this is done, the Worker: Architect identifies a number of classes that should be considered as architecturally significant, and includes these classes in the logical view section of the Artifact: Software Architecture Document. The resulting analysis artifacts are then reviewed.
Design: Adjust to the implementation environment, decide how the "drivers" are to be designed, and refine the definition of classes, packages and subsystems; inspect results.	The Worker: Architect then refines the architecture by deriving the design mechanisms (e.g. programming language, database, distribution mechanism, communication mechanism) needed by the earlier identified analysis mechanisms (see Activity: Identify Design Mechanisms in the Analysis & Design Workflow). Artifact: Design Subsystems are defined and design classes are allocated to them; the interfaces to subsystems are identified. Remaining design classes are partitioned into packages, and responsibilities for subsystems and packages are allocated to Worker: Designers. Instances of classes and subsystems are used by designers to describe the realizations of the selected use cases and/or scenarios (see Workflow Detail: Design Components in the Analysis & Design workflow). This puts requirements on the employed model elements and their associated design mechanisms; in the process the interaction diagrams previously created are refined. The requirements put on each design mechanism are handled by the architect (revisit Activity Identify Design Mechanisms in the Analysis & Design workflow). The logical view is updated accordingly by the architect. The resulting design artifacts are then reviewed.

Consider the concurrency and distribution aspect of the architecture.	The next step for the architect is to consider the concurrency and distribution required by the system. This is done by studying the tasks and processes required and the physical network of processors and other devices (see Activity: Describe the Run-time Architecture and Activity: Describe Distribution in the Analysis & Design Workflow). An important input to the architect here are the designed use cases in terms of collaborating objects in interaction diagrams; these objects are allocated to tasks and processes, which in turn are allocated to processors and other devices. This results in both a logical and physical distribution of functionality.
Inspect the architecture	The architecture is reviewed. See Activity: Review the Architecture.
Implementation: Consider the physical packaging of the architecture.	An Worker: Architect now considers the impact of the architectural design onto the implementation model, and defines the initial structure of the implementation model (revisit Activity: Structure the Implementation Model in the Analysis & Design workflow).
Implementation: Plan the integration.	A system integrator now studies the use cases that are to be implemented in this iteration, and defines the order in which subsystems should be implemented, and later integrated into an architectural prototype (see Workflow Detail: Plan the Integration within an Iteration in the Implementation workflow). The results of this planning should be reflected in the Artifact: Software Development Plan.
Test: Plan integration tests and system tests.	A test designer now plans the system tests and the integration tests. Selects measurable testing goals to be used when assessing the architecture. These goals could be expressed in terms of being able to execute a use-case scenario, with a certain response time, or response time under specified load. The test designer also identifies and implements test cases and test procedures. (See Workflow Detail: Plan and Design Test and Workflow Detail: Plan and Design System Test in the Test workflow.)
Implementation: Implement the classes and integrate.	A number of implementers now implement and unit test the classes identified in the architectural design (Step 5, 6, and 7). The implementations of the classes are physically packaged into components and subsystems in the implementation model. The implementers also fix defects (see Workflow Detail: Implement Classes Within an Iteration in the Implementation workflow). The testers integration test the implementation subsystem (see Workflow Detail: Execute Integration Test in the Test workflow), and then the implementers release the tested implementation subsystems to system integration.
Integrate the implemented parts.	The system integrators incrementally integrate the subsystems into an executable architectural prototype (see Workflow Detail: Integrate the System within an Iteration in the Implementation workflow). Each build is tested (see Workflow Detail: Execute Integration Test in the Test workflow).
Test: Assess the executable architecture.	Once the whole system (as defined by the goal of this iteration) has been integrated, the system tester tests the system (see Workflow Detail: Execute System Test in the Test workflow). The test designer then analyzes the results of the test to make sure the testing goals have been reached (see Workflow Detail: Evaluate System Test in the Test workflow). The Worker: Architect will then assess this result and compare it with the risk initially identified.

Assess the iteration itself.	Lastly, the Worker: Project Manager compares the iteration's actual cost, schedule, and content with the iteration plan; determine if rework needs to be done, and if so, assign to future iterations; update the risk list (see Artifact: Risk List); update the project plan (see Artifact: Software Development Plan); and prepare an outline of an iteration plan for the next iteration (see Artifact: Iteration Plan). Productivity figures, size of code, and size of database might be interesting to consider here. The Worker: Project Manager, in cooperation with the Worker: Process Engineer and the Worker: Tool Specialist, evaluate the use process and tools.

Result

The result of this initial iteration would be a first cut at the architecture, consisting of fairly described architectural views (use-case view, logical view, process view, deployment view, implementation view, and an executable architecture prototype.

Subsequent Iterations In The Elaboration Phase

Subsequent iterations can be initiated to further enhance the understanding of the architecture. This might imply a further enhancement of the design or implementation model (that is, the realization of more use cases, in priority order, of course). Whether this needs to take place depends on considerations such as the complexity of the system and its architecture, associated risks, and domain experience.

In each iteration the supporting environment is further refined. If the first Elaboration iteration focused on preparing the environment for Analysis & Design, and Implementation, then the second iteration may focus on preparing the test environment. Preparing the test environment includes configuring the test process, and writing that part of the development case, preparing templates and guidelines for test and setting up the test tools.

Phase 3: Construction

Objectives

The goal of the construction phase is on clarifying the remaining requirements and completing the development of the system based upon the baselined architecture. The construction phase is in some sense a manufacturing process, where emphasis is placed on managing resources and controlling operations to optimize costs, schedules, and quality. In this sense the management mindset undergoes a transition from the development of intellectual property during inception and elaboration, to the development of deployable products during construction and transition.

The primary objectives of the construction phase include:

- Minimizing development costs by optimizing resources and avoiding unnecessary scrap and rework.
- Achieving adequate quality as rapidly as practical
- Achieving useful versions (alpha, beta, and other test releases) as rapidly as practical
- Completing the analysis, design, development and testing of all required functionality.
- To iteratively and incrementally develop a complete product that is ready to transition to its user community. This implies describing the remaining use cases and other requirements, fleshing out the design, completing the implementation, and testing the software.
- To decide if the software, the sites, and the users are all ready for the application to be deployed.
- To achieve some degree of parallelism in the work of development teams. Even on smaller projects, there are typically components that can be developed independently of one another, allowing for natural parallelism between teams (resources permitting). This parallelism can accelerate the development activities significantly; but it also increases the complexity of resource management and workflow synchronization. A robust architecture is essential if any significant parallelism is to be achieved.

Essential Activities

- Resource management, control and process optimization
- Complete component development and testing against the defined evaluation criteria
- Assessment of product releases against acceptance criteria for the vision.

Milestone: Initial Operational Capability

At the Initial Operational Capability Milestone, the product is ready to be handed over to the Transition Team. All functionality has been developed and all **alpha** testing (if any) has been completed. In addition to the software, a user manual has been developed, and there is a description of the current release.

Evaluation Criteria

The evaluation criteria for the construction phase involve the answers to these questions:

- Is this product release stable and mature enough to be deployed in the user community?
- Are all the stakeholders ready for the transition into the user community?
- Are actual resource expenditures versus planned still acceptable?

Transition may have to be postponed by one release if the project fails to reach this milestone.

Artifacts

Essential Artifacts (in order of importance)	State at milestone
"The System"	The executable system itself, ready to begin "beta" testing.
Deployment Plan	Initial version developed, reviewed and baselined.
Implementation Model (and all constituent artifacts, including Components)	Expanded from that created during the elaboration phase; all components created by the end of the construction phase.
Test Model (and all constituent artifacts)	Tests designed and developed to validate executable releases created during the construction phase.
Training Materials	User Manuals & other training materials. Preliminary draft, based on use cases. May be needed if the system has a strong user interface aspect.
Iteration Plan	Iteration plan for the transition phase completed and reviewed.
Design Model (and all constituent artifacts)	Updated with new design elements identified during the completion of all requirements.
Development Case	Refined based on early project experience. The development environment, including the process, tools and automation support required to support the transition team will have been put in place.
Project-Specific Templates	The document templates used to develop the document artifacts.
Tools	The tools used to support the work in Construction are installed.
Data Model	Updated with all elements needed to support the persistence implementation (e.g. tables, indexes, object-to-relational mappings, etc.)
Optional Artifacts	**State at milestone**
Supplementary Specifications	Updated with new requirements (if any) discovered during the construction phase.
Use-Case Model (Actors, Use Cases)	Updated with new use cases (if any) discovered during the construction phase.

Sample Iteration Plan: Construction Phase

This illustration shows the relationship of the workflows in an early construction iteration. It is constructed from the Workflow Details as they would appear at that time. The intent is to indicate dependencies and show where workflows occur in parallel. The lengths of the bars in the chart (indicating duration) have no absolute significance. For example, it is not intended to convey that Plan the Integration and Plan Test must have the same duration. There is also no intention to suggest the application of a uniform level of effort across the duration of the workflows. An indication of the relative effort can be seen in the Process Overview found in Appendix B. Figure 6 gives the construction phase workflow details.

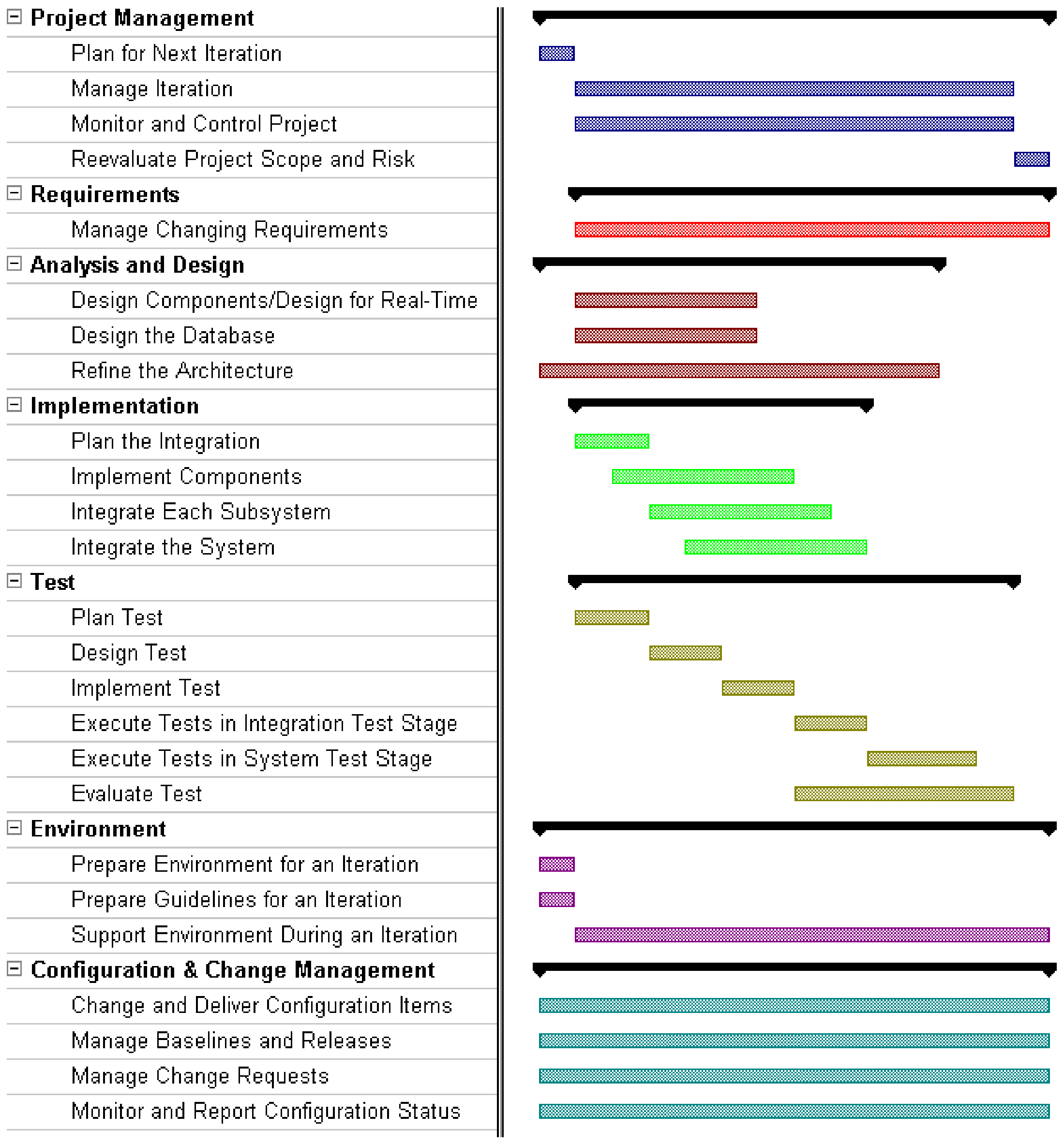


Figure 6: Construction Phase Workflow Detail

Note that there is significant continuing design work shown in this iteration, indicating that it is early in the construction cycle. In later construction iterations, this will diminish as design work completes, when the design work remaining will relate to Change Requests (defects and enhancements) that impact design. Requirements discovery and refinement is shown as complete at this stage, the remaining effort relating entirely to the management of change.

Workflow

Project Management: Plan the Iteration.	The project manager updates the iteration plan based on what new functionality is to be added during the new iteration, factoring in the current level of product maturity, lessons learned from the previous iterations, and any risks that need to be mitigated in the upcoming iteration (see Artifact: Iteration Plan and Artifact: Risk List).
Environment: Prepare the environment for the iteration.	Based on the evaluation of process and tools in the previous iteration, the Worker: Process Engineer further refines the development case, templates and guidelines. The Worker: Tool Specialist does the necessary changes to the tools.
Implementation: Plan system-level integration.	Integration planning takes into account the order in which functional units are to be put together to form a working/testable configuration. The choice depends on the functionality already implemented, and what aspects of the system need to be in place to support the overall integration and test strategy. This is done by the system integrator (see Workflow Detail: Plan the Integration within an Iteration in the Implementation workflow), and the results are documented in the Artifact: Integration Build Plan. The Integration Build Plan defines the frequency of builds and when given 'build sets' will be required for ongoing development, integration, and test.
Test: Plan and design system-level test.	The test designer ensures that there will be an adequate number of test cases and procedures to verify testable requirements (see Workflow Detail: Plan and Design Test in the Test workflow). The test designer must identify and describe the test cases, and identify and structure the test procedures. In general, each test case will have at least one associated test procedure. The test designer should review the accumulated body of tests from preceding iterations, which could be modified and therefore available for re-use in regression testing for the current and future iteration builds.
Analysis & Design: Refine Use-Case Realizations.	Designers refine the model elements identified in previous iterations by allocating responsibilities to specific model elements (classes or subsystems) and updating their relationships and attributes. New elements may also need to be added to support possible design and implementation constraints (see Workflow Detail: Design Components) Changes to elements may require changes in package and subsystem partitioning (see Activity: Incorporate Existing Design Elements). Results of the analysis need to be followed by review(s).
Test: Plan and design integration tests at the subsystem and system level.	Integration tests focus on how well the developed components interface and function together. The test designer needs to follow the test plan that describes the overall test strategy, required resources, schedule, and completion and success criteria. The designer identifies the functionality that will be tested together, and the stubs and drivers that will need to be developed to support the integration tests. The implementer develops the stubs and drivers based on the input from the test designer (see Workflow Detail: Implement Test in the Test workflow).
Implementation: Develop Code and Test Unit	Implementers develop code, in accordance with the project's programming guidelines, to implement the Artifact: Components in the implementation model. They fix defects and provide any feedback that may lead to design changes based on discoveries made in implementation (see Workflow Detail: Implement Classes Within an Iteration in the Implementation workflow).

Implementation: Plan and Implement Unit Tests.	The implementer needs to design unit tests so that they address what the unit does (black-box), and how it does it (white-box). Under black-box (specification) testing the implementer needs to be sure that the unit, in its various states, performs to its specification, and can correctly accept and produce a range of valid and invalid data. Under white-box (structure), testing the challenge for the implementer is to ensure that the design has been correctly implemented, and that the unit can be successfully traversed through each of its decision paths (see Workflow Detail: Implement Classes Within an Iteration in the Implementation workflow).
Implementation: Test Unit within Subsystem.	Unit Test focuses on verifying the smallest testable components of the software. Unit tests are designed, implemented, and executed by the implementer of the unit. The emphasis of unit test is to ensure that white-box testing produce the expected results, and that the unit conforms to the project's adopted quality and development standards.
Implementation and Test: Integrate Subsystem.	The purpose of subsystem integration is to combine units that may come from many different developers within the subsystem (part of the implementation model), into an executable 'build set'. The implementer in accordance with the plan integrates the subsystem by bringing together completed and stubbed classes that constitutes a build (see Workflow Detail: Integrate Each Subsystem Within an Iteration in the Implementation workflow). The implementer integrates the subsystem incrementally from the bottom-up based on the compilation-dependency hierarchy.
Implementation: Test Subsystem.	Testers execute test procedures developed in accordance with activities identified in Steps 3 and 5 (see Workflow Detail: Execute Integration Test in the Test workflow). If there are any unexpected test results, the testers log the defects for arbitration on when they are to be fixed.
Implementation: Release Subsystem.	Once the subsystem has been sufficiently tested and it is ready for integration at the system level, the implementer 'releases' the tested version of the subsystem from the team integration area into an area where it becomes visible, and usable, for system-level integration.
Implementation: Integrate System.	The purpose of system integration is to combine the currently available implementation model functionality into a build. The system integrator incrementally adds subsystems, and creates a build that is handed over to testers for overall integration testing (see Workflow Detail: Integrate the System within an Iteration in the Implementation workflow).
Test: Test Integration	Testers execute test procedures developed in accordance with activities identified in Steps 3 and 5. The testers execute integration tests and review the results. If there are any unexpected results, the testers log the defects (see Workflow Detail: Execute Integration Test in the Test workflow).
Test: Test System	Once the whole system (as defined by the goal of this iteration) has been integrated, it is subjected to system testing (see Workflow Detail: Execute System Test in the Test workflow). The test designer will then analyze the results of the test to make sure the testing goals have been reached (see Workflow Detail: Evaluate System Test in the Test workflow).
Project Management: Assess the iteration itself.	Lastly, the project manager compares the iteration's actual cost, schedule, and content with the iteration plan; determine if rework needs to be done, and if so, assign to future iterations; update the risk list (see Artifact: Risk List); update the project plan (see Artifact: Software Development Plan); prepare an outline of an iteration plan for the next iteration (see Artifact: Iteration Plan). Productivity figures, size of code, or and size of database might be interesting to consider here. The project manager, in cooperation with the process engineer and the tool specialist, evaluates the process and the use of tools. These lessons-learned will be used when preparing the environment for the following iteration.

Result

The main result of a late iteration in the construction phase is that more functionality is added, which yields an increasingly more complete system. The results of the current iteration are made visible to developers to form the basis of development for the subsequent iteration.

Phase 4: Transition

Objectives

The focus of the Transition Phase is to ensure that software is available for its end users. The Transition Phase can span several iterations, and includes testing the product in preparation for release, and making minor adjustments based on user feedback. At this point in the lifecycle, user feedback should focus mainly on fine tuning the product, configuring, installing and usability issues, all the major structural issues should have been worked out much earlier in the project lifecycle.

By the end of the Transition Phase lifecycle objectives should have been met and the project should be in a position to be closed out. In some cases, the end of the current life cycle may coincide with the start of another lifecycle on the same product, leading to the next generation or version of the product. For other projects, the end of Transition may coincide with a complete delivery of the artifacts to a third party who may be responsible for operations, maintenance and enhancements of the delivered system.

This Transition Phase ranges from being very straightforward to extremely complex, depending on the kind of product. A new release of an existing desktop product may be very simple, whereas the replacement of a nation's air-traffic control system may be exceedingly complex.

Activities performed during an iteration in the Transition Phase depend on the goal. For example, when fixing bugs, implementation and test are usually enough. If, however, new features have to be added, the iteration is similar to one in the construction phase requiring analysis&design, etc.

The Transition Phase is entered when a baseline is mature enough to be deployed in the end-user domain. This typically requires that some usable subset of the system has been completed with acceptable quality level and user documentation so that transitioning to the user provides positive results for all parties.

The primary objectives of the Transition Phase are:

- beta testing to validate the new system against user expectations
- beta testing and parallel operation relative to a legacy system that it's replacing
- converting operational databases
- training of users and maintainers
- roll-out to the marketing, distribution and sales forces
- deployment-specific engineering such as cutover, commercial packaging and production, sales roll-out, field personnel training
- tuning activities such as bug fixing, enhancement for performance and usability

- assessment of the deployment baselines against the complete vision and the acceptance criteria for the product
- achieving user self-supportability
- achieving stakeholder concurrence that deployment baselines are complete
- achieving stakeholder concurrence that deployment baselines are consistent with the evaluation criteria of the vision

Essential activities

- executing deployment plans
- finalizing end-user support material
- testing the deliverable product at the development site
- creating a product release
- getting user feedback
- fine-tuning the product based on feedback
- making the product available to end users

Milestone: Product Release

At the end of the transition phase is the fourth important project milestone, the **Product Release Milestone**. At this point, you decide if the objectives were met, and if you should start another development cycle. In some cases this milestone may coincide with the end of the inception phase for the next cycle. The Product Release Milestone is the result of successful completion of the Activity: Project Acceptance Review.

Evaluation Criteria

The primary evaluation criteria for the transition phase involve the answers to these questions:

- Is the user satisfied?
- Are actual resources expenditures versus planned expenditures acceptable?

At the Product Release Milestone, the product is in production and the post-release maintenance cycle begins. This may involve starting a new cycle, or some additional maintenance release.

Artifacts

Essential Artifacts (in order of importance)	**State at milestone**
The Product Build	Complete in accordance with the product requirements. The final product should be useable by the customer.
Release Notes	Complete.
Installation Artifacts	Complete.
Training Material	Complete to ensure that the customer can become self-sufficient in the use and maintenance of the product.
End-User Support Material	Complete to ensure that the customer can become self-sufficient in the use and maintenance of the product.

Optional Artifacts	State at milestone
Test Model	The test model may be provided in the situation where the customer wants to runs on-site testing.
'Shrink-wrap' Product Packaging	In the case of creating a shrink-wrap product, the contractor will need the necessary packaging artifacts to help retail the product.

Sample Iteration Plan: Transition Phase

This illustration shows the relationship of the workflows in an iteration late in the transition phase - if the Workflow Detail 'Close-Out Project' is invoked, then this would be the final iteration. It is constructed from the Workflow Details as they would appear at that time. The intent is to indicate dependencies and show where workflows occur in parallel. The lengths of the bars in the chart (indicating duration) have no absolute significance. For example, it is not intended to convey that Refine the Architecture and Acceptance Test the Product (at the development site) have the similar duration. There is also no intention to suggest the application of a uniform level of effort across the duration of the workflows. An indication of the relative effort can be seen in the Process Overview found in Appendix B. Figure 7 gives the detail workflow for the transition phase.

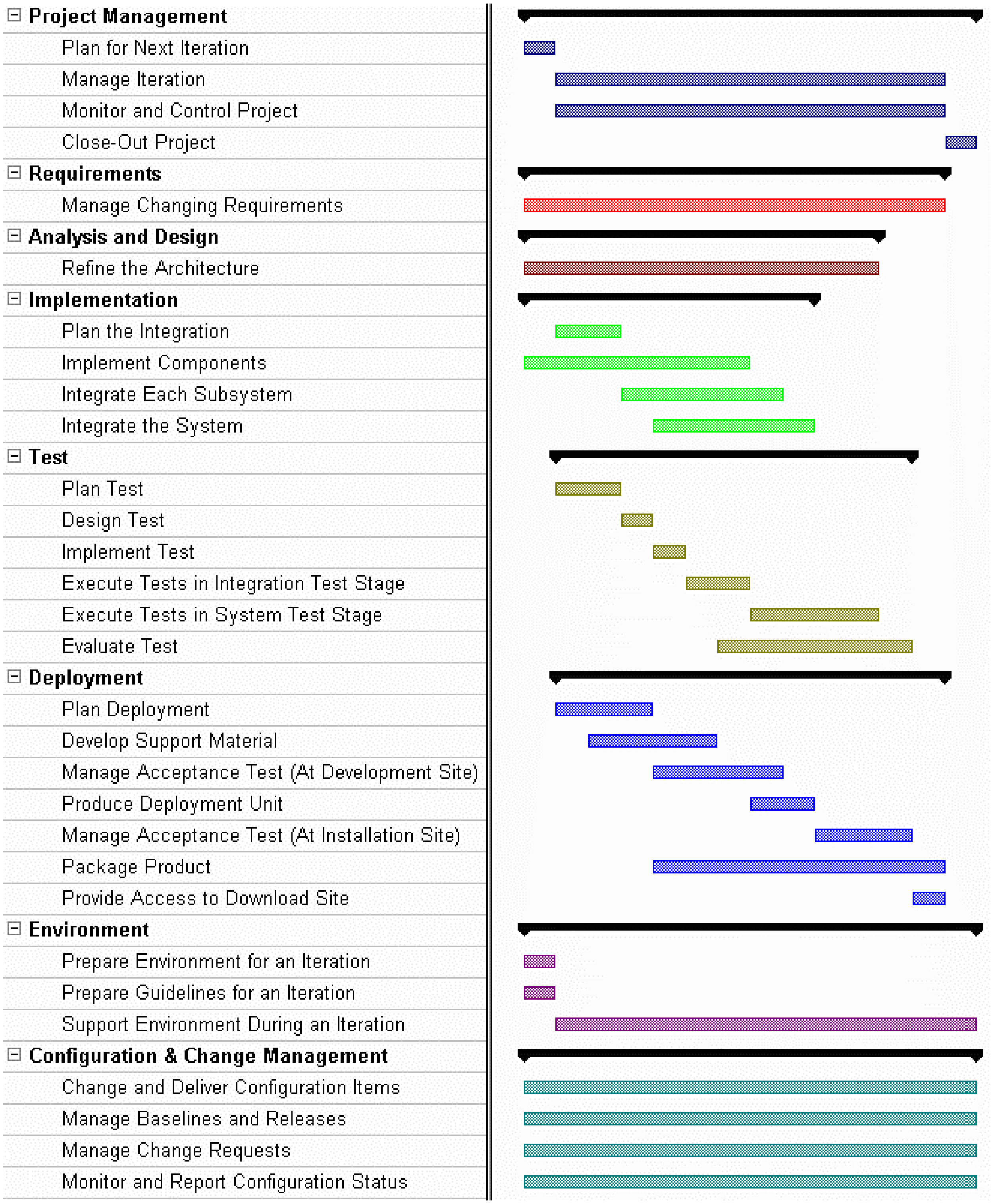


Figure 7: Transition Phase Workflow Detail

Workflow

Project Management	Late in the Transition Phase, the main driver for planning in Activity: Develop Iteration Plan is the delivery of reliable software, with acceptable performance and complete functionality, to the customer. Accordingly, Change Requests (mainly defects and feedback from beta testing) are the Project Manager's major planning input for continuing development. Based on the number and severity of the Change Requests, the Project Manager may invoke risk management activities (through the Artifact: Risk List), for example in the management of changing requirements, or architecture refinement. The Project Manager has also to plan for the production of end-user support and installation material, and the contractually formal aspects of acceptance test. The Project Manager initiates the iteration in Activity: Initiate Iteration, then monitors and reports on project status in Workflow Detail: Monitor and Control Project. At completion, the results of the iteration are examined in Activity: Assess Iteration, and if this is the final iteration, the project manager prepares the project for shutdown.
Requirements and Analysis & Design	Given the nature of the iterative development process, it is expected that the requirements will be very stable, if not completely frozen, by this time. Even so, some feedback that affects system requirements, or their interpretation, should be anticipated and the impact of this on scope has to be understood and controlled in Workflow Detail: Manage Changing Requirements. It is important that the system not be allowed to change in an ad hoc way during transition. Equally, the objective of analysis and design in this phase, in Workflow Detail: Refine the Architecture, is to maintain architectural integrity and perform the necessary run-time tuning and physical distribution adjustments to meet requirements for performance, capacity, and reliability.
Implementation	The planning for implementation during transition in Workflow Detail: Plan the Integration is driven by the feedback from beta test and other Change Requests raised during test by the project itself. As defects are fixed in Activity: Fix a Defect, and subsystems mature, they are integrated into builds for testing. In transition, the main work is in fixing defects in components, not adding new components. Unit testing (in Activity: Perform Unit Tests) is still required, but the purpose in transition is to verify changes and avoid regression, not complete functional verification. In subsystem and system integration during transition, (in Workflow Details: Integrate Each Subsystem and Integrate the System), completed components are available, so the use of 'stubs' is unnecessary, and again the purpose is to verify and validate changes and check for regressions. It is not usually necessary to perform integration in the piecewise fashion used during construction because the interfaces are stable by this time, and the Integrator can take a more optimistic approach.
Test	As with unit tests, the focus for subsystem and system level tests during transition is on the verification and validation of changes and the avoidance of regression. In addition, there will often be a requirement for formal acceptance testing, which usually involves a repeat of all or part of the system level tests. The planning for test during transition (in Workflow Detail: Plan Test) thus has to provide effort and resources for some level of continued test design and implementation in Workflow Details: Design Test and Implement Test (because there will be changes during transition); regression testing, for which the effort and resources will depend on the chosen approach (for example, retest everything, retest to an operational profile, or retest changed software), and acceptance testing, which should not require the development of new tests. As defects are fixed and beta feedback incorporated, successive builds are tested (in Workflow Details: Execute Tests in Integration Test Stage and Execute Tests in System Test Stage) until the system is deemed fit to undergo acceptance testing (usually through a repeat of all or part of the system level tests in Workflow Detail: Execute Tests in

	System Test Stage). In transition, particularly during acceptance testing, the Customer, Test Designer and Deployment Manager will collaborate during Workflow Detail: Evaluate Test, to decide which test results are acceptable, whether to continue testing, and which tests must be repeated.
Deployment	Deployment Planning (in Workflow Detail: Plan Deployment) at this stage in transition is concerned with establishing the schedule and resources (in the Artifact: Deployment Plan) for (continued) development of end-user support material, acceptance testing, and production, packaging and distribution of software deployment units. Beta testing has been completed in previous iterations in transition. The Deployment Manager also produces the Artifact: Bill of Materials in this workflow detail. Any remaining work to produce the Artifact: End-User Support Material (for example, user guides, operational guides, maintenance guides) and the Artifact: Training Materials is completed by the Workers: Technical Writer and Course Developer respectively, in Workflow Detail: Develop Support Material. Once the system is deemed fit, acceptance testing commences, managed by the Deployment Manager in Activity: Manage Acceptance Test. After successful testing at the development site, the Deployment Manager initiates the production of the deployment units (for installation at the customer's site), by producing the Artifact: Release Notes. These and the Artifact: Installation Artifacts, produced by the Worker: Implementer, are input (with others) to the Activity: Create Deployment Unit (in the Configuration Management workflow). Frequently, at least a portion of acceptance testing is performed at the customer's site, usually after initial acceptance testing at the development site. In parallel with acceptance testing, the artwork for the product packaging is developed by the Worker: Graphic Artist in Activity: Create Product Artwork. Finally, the deployment manager initiates the production of the product for distribution in Activity: Release to Manufacturing, and quality checks the result in Activity: Verify Manufactured Product, before the product is shipped.
Environment	There should be little or no development work to be done on the environment by this stage, the work during transition should be almost wholly support and maintenance, in the Workflow Detail: Support Environment During an Iteration.
Configuration Management	The configuration management activities continue in parallel with the remaining implementation and test with increasing emphasis on the formality of change control. The Artifact: Deployment Unit is created in Workflow Detail: Manage Baselines and Releases, by the Configuration Manager, as a precursor to final product packaging. All requests for change will require sanction by a project-level CCB (and the customer) during transition, as part of Workflow Detail: Manage Change Requests. Finally, as part of acceptance, it is usually necessary to do a Functional Configuration Audit (FCA) and a Physical Configuration Audit (PCA) in Activity: Perform Configuration Audits.

Result

This final iteration in the transition phase culminates in the delivery to the customer of a complete system (and ancillary support artifacts) with functionality and performance as specified, and demonstrated in acceptance testing. The customer takes ownership of the software after a successful acceptance test.

APPENDICES

APPENDIX A

Activity: Lifecycle Milestone Review

Purpose • To review the state of the project at the end of a phase, and determine whether the project should proceed to the next phase.	
Steps • Schedule Lifecycle Milestone Review Meeting • Distribute Meeting Materials • Conduct Lifecycle Milestone Review Meeting • Record Decision	
Input Artifacts: • Iteration Assessment • Status Assessment • Software Development Plan	**Resulting Artifacts:** • Review Record
Frequency: Once per phase	
Worker: Project Reviewer	

Workflow Details: • Core Workflow: Project Management   • close-out Phase

A Lifecycle Milestone Review is held at the conclusion of each phase to determine, following the completion of the final iteration of the phase, whether the project should be allowed to proceed to the next phase. It marks a point at which management and technical expectations should be resynchronized, but the issues to be considered should relate mainly to the management of the project - major technical issues should have been resolved with the final iteration (of the phase), and in the subsequent Activity: Prepare for Phase Close-Out.

A review is held at each of the major milestones, in particular at:

- the Lifecycle Objectives Milestone at the end of the **Inception Phase**
- the Lifecycle Architecture Milestone at the end of the **Elaboration Phase**
- the Initial Operational Capability Milestone at the end of the **Construction Phase**
- the Product Release Milestone at the end of the **Transition Phase**

Issues for Consideration

The issues to be considered are, by default, those canvassed in the Status Assessment, e.g.:

- has the project made adequate progress (in delivering capability, quality and planned artifacts) across the phase?
- is the project's risk profile acceptable to enter the next phase?
- is the project's scope well-understood and acceptable to all stakeholders?
- are the project's baselines in a known state according to configuration audits?

- has the project performed acceptable on cost and schedule?

Financial considerations will be particularly important if the phase end also marks the end of a contract.

Schedule Lifecycle Milestone Review Meeting

The Lifecycle Milestone Review meeting is a meeting between a customer representative(s), the project's management team (the project manager, plus the team leads for the various functional areas of the project team), and the Project Review Authority.

Once the attendees of the meeting have been identified, set a date/time for the meeting to take place. It is important that sufficient lead time is allowed for the participants to review the materials that will be used as the basis for the approval decision.

Distribute Meeting Materials

Prior to the meeting, distribute the review materials to the reviewers. Make sure these materials are sent out sufficiently in advance of the meeting to allow the reviewers adequate time to review them.

Conduct Lifecycle Milestone Review Meeting

During the meeting, the attendees will be mainly concerned with the Status Assessment. See the Issues for Consideration.

At the end of the meeting, the reviewers should make the decision to approve or not. If the remaining issues are few and relatively minor, the customer may decide to accept the product conditionally upon certain corrective actions being taken. In this situation the Project Manager may choose to initiate a new iteration to deal with the issues arising, depending on their significance, or simply deal with issues as an extension of the final iteration, the difference being in the amount of planning needed. If the results of the phase are found to be unacceptable, the Project Manager may be obliged to initiate another iteration, or perhaps the resolution of the problem is taken out of the Project Manager's hands, and left to the customer and the Project Review Authority.

The result of the Lifecycle Milestone Review Meeting can be one of the following:

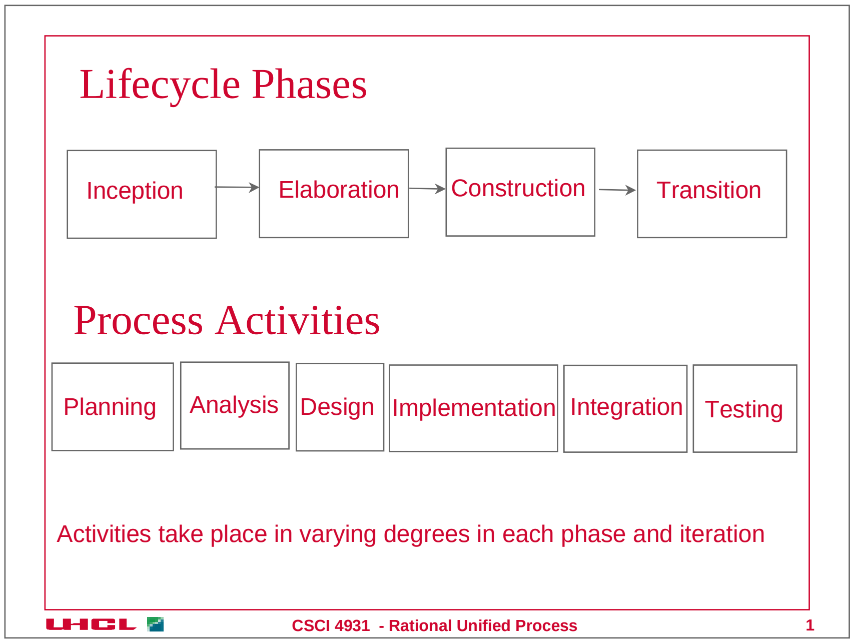
Phase Accepted	The customer representative agrees that the project has met expectations for the phase, and can proceed to the next phase.
Conditional Acceptance	The customer representative agrees that the project may proceed to the next phase, subject to the completion of specified corrective actions.
Phase Not Accepted	The project has failed to achieve the expectations for the phase: either a further iteration is scheduled, or the various stakeholders have recourse to the contract, to re-scope or terminate the project.

Record Decision

At the end of the meeting, a Review Record is completed, capturing any important discussions or action items, and recording the results of the Lifecycle Milestone Review. If the result was "not accepted", a follow-up review should be tentatively scheduled - if the project is allowed to continue. A firmer date will be set following the planning for the additional iteration.

APPENDIX B

OVERVIEW



Lifecycle Phases

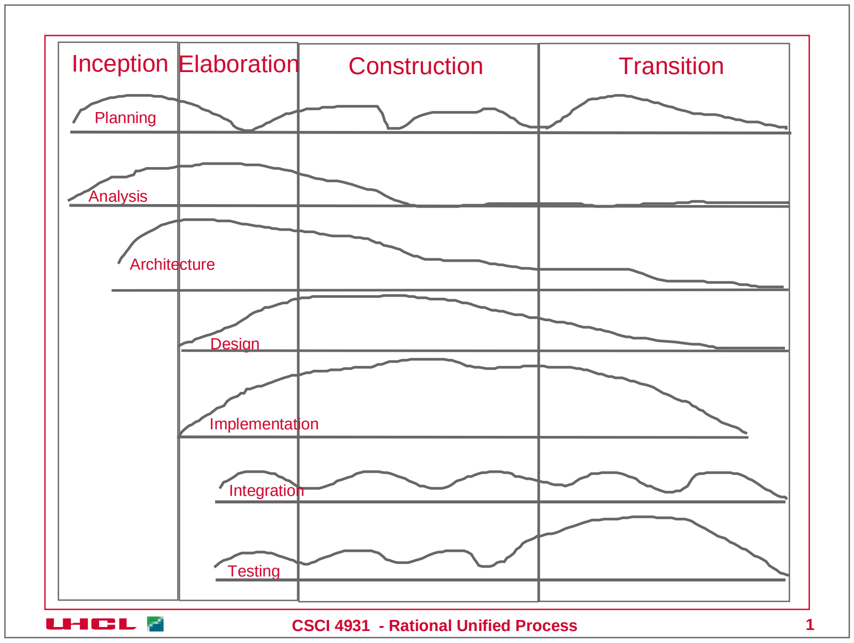
Inception → Elaboration → Construction → Transition

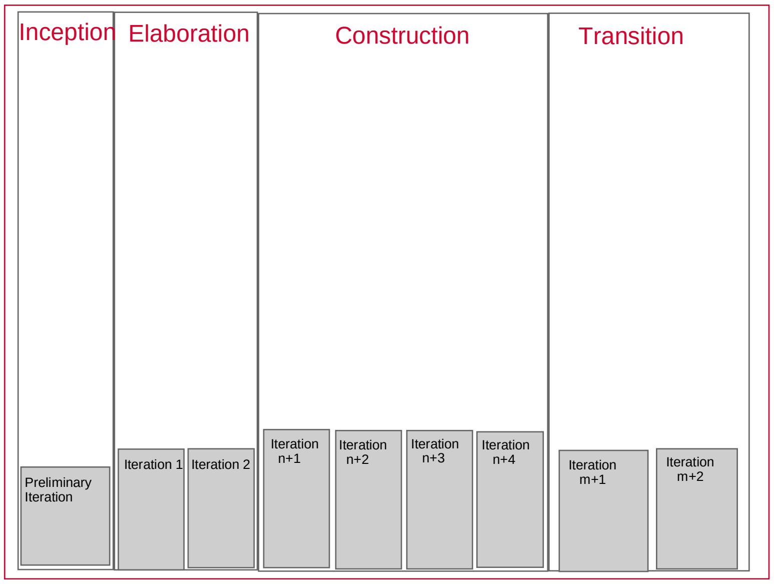
Process Activities

Planning | Analysis | Design | Implementation | Integration | Testing

Activities take place in varying degrees in each phase and iteration

UHCL CSCI 4931 - Rational Unified Process 1




Inception
Elaboration
Construction
Transition
Preliminary Iteration
Iteration 1
Iteration 2
Iteration n+1
Iteration n+2
Iteration n+3
Iteration n+4
Iteration m+1
Iteration m+2