.

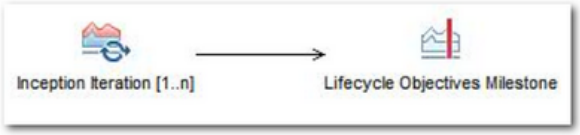
.

2.2 #phase# Inception Phase

- #iteration# Inception Iteration [1..n]
- 2.2.1 #activity# Initiate Project
  - 2.2.1.1 #task# Develop Technical Vision
  - 2.2.1.2 #task# Plan Project
- 2.2.2 #activity# Plan and Manage Iteration
  - 2.2.2.1 #task# Plan Iteration
  - 2.2.2.2 #task# Manage Iteration
  - 2.2.2.3 #task# Assess Results
- 2.2.3 #activity# Prepare Environment
  - 2.2.3.1 #task# Tailor the Process
  - 2.2.3.2 #task# Set Up Tools
  - 2.2.3.3 #task# Verify Tool Configuration and Installation
  - 2.2.3.4 #task# Deploy the Process

- 2.2.4 #activity# Identify and Refine Requirements
  - 2.2.4.1 #task# Identify and Outline Requirements
  - 2.2.4.2 #task# Detail Use-Case Scenarios
  - 2.2.4.3 #task# Detail System-Wide Requirements
  - 2.2.4.4 #task# Create Test Cases
- 2.2.5 #activity# Agree on Technical Approach
  - 2.2.5.1 #task# Envision the Architecture

WBS

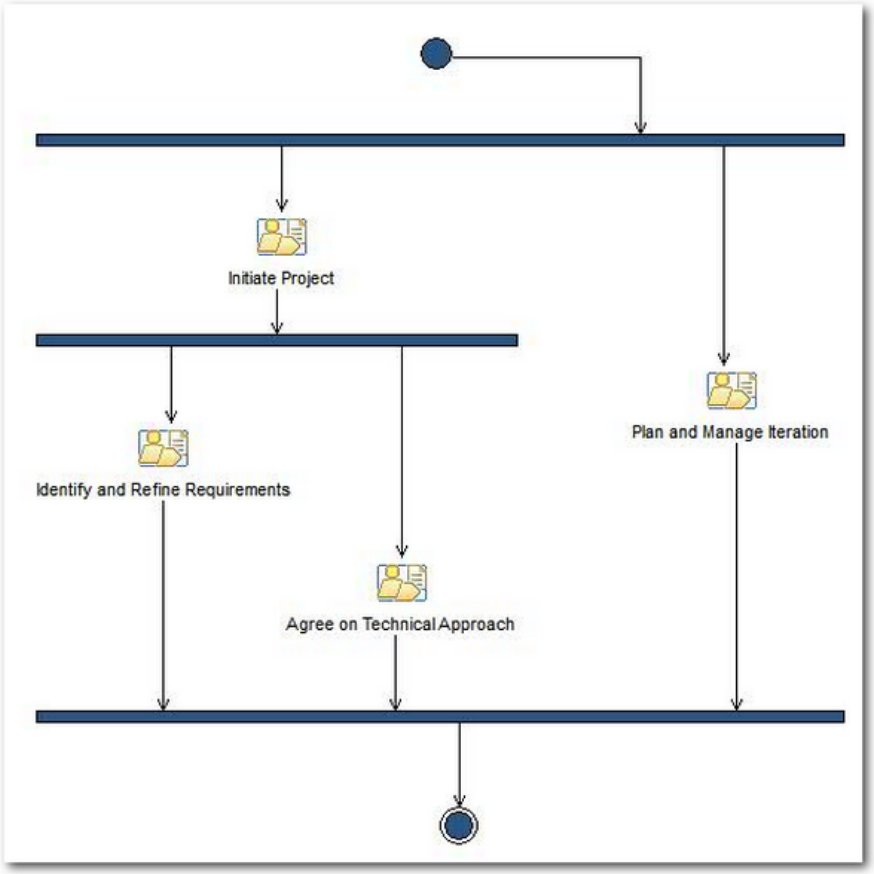


.

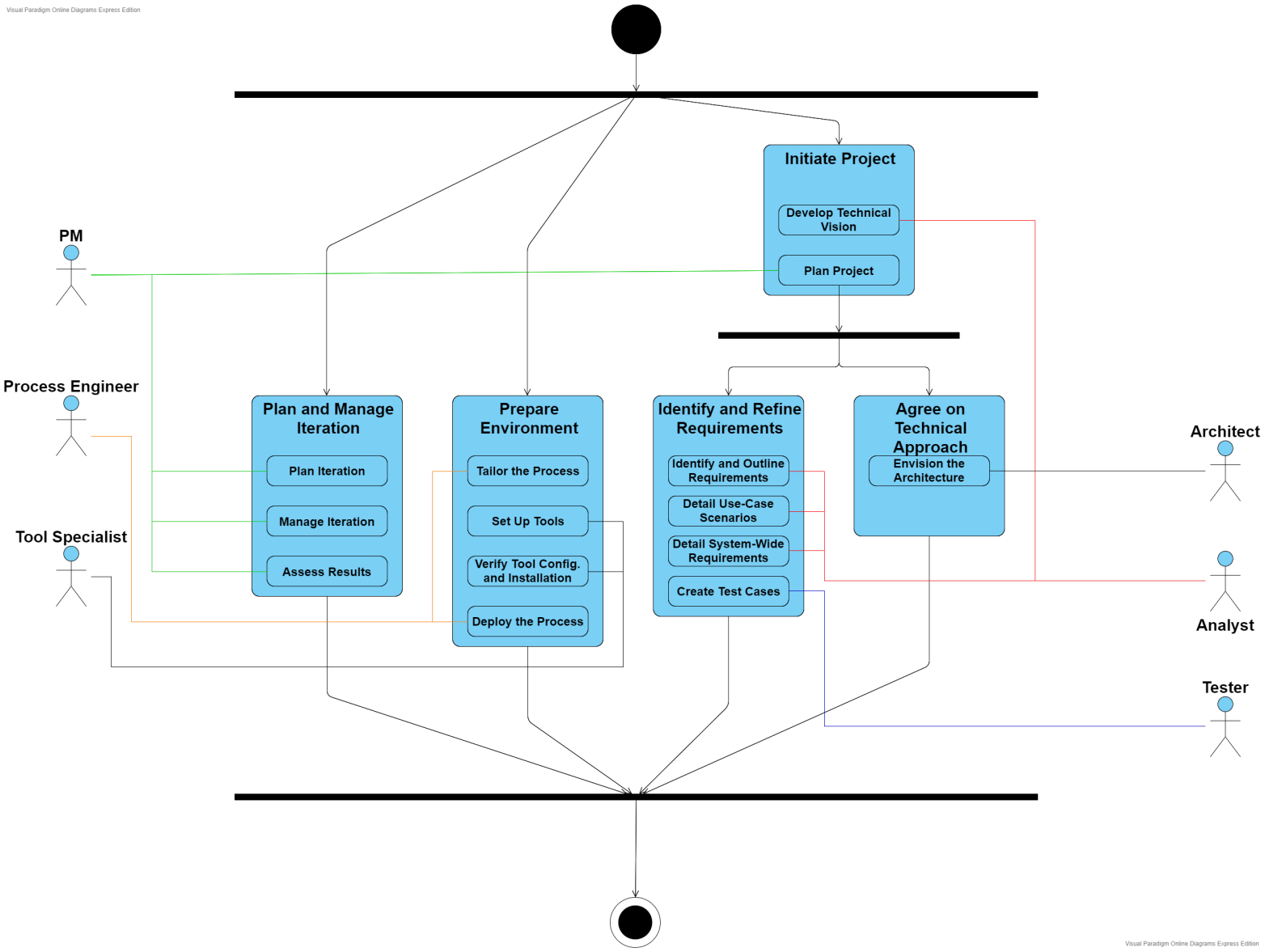
.

#iteration# Inception Iteration [1..n]

WBS



Visual Paragigm Online Link



Description

This iteration template defines the activities (and associated roles and work products) performed in a typical iteration in the Inception phase.

The project starts with the assumption that you have:

- Created the business case
- Identified the project manager
- Defined the team (at least for the first iteration)
- Put in place the development environment (including tools and infrastructure)
- Prepared your organization to follow the delivery process suggested by the published process description

The number and length of each Inception iteration will vary depending upon the needs of the project.

The following table summarizes the Inception phase objectives and what activities address each objective:

Inception phase objectives and activities

Phase objectives	Activities that address objectives

Understand what to build	• **Identify and Refine Requirements**
Identify key system functionality	• **Initiate Project,** • **Identify and Refine Requirements**
Determine at least one possible solution	• **Agree on Technical Approach**
Understand the cost, schedule, and risks associated with the project	• **Initiate Project,** • **Plan and Manage Iteration**

.

.

2.2.1 #activity# Initiate Project

Summary
Kick off the project and gain agreement with the stakeholders on the scope of the project, and an initial plan for achieving it. This activity bundles tasks required to define the vision and create a project plan.

WBS

- #task# Develop Technical Vision
- #task# Plan Project

Description
This activity takes place at the beginning of the first iteration, when the project starts. The goal of this activity is **to establish the vision** for both the project and the project plan at a high level of granularity.

The people in the roles involved at this time collaborate to perform this activity:

- The **Analyst** and **Stakeholder** roles work together to define the vision for the project, capturing the customer needs and features for the system under development. Needs and features are captured to the extent required to reach agreement about the scope of the project.
- The **project manager** (collaborating and reaching agreement with the development team and stakeholders) proposes a high-level project plan that includes the milestones for the four phases, and time-boxed iterations for each phase. This plan, along with the allocation of staff to the project, evolves throughout the phases and defines the pace of the project.

.

.

2.2.1.1 #task# Develop Technical Vision

Summary
Define the vision for the future system. Describe the problem and features based on stakeholder requests.

Purpose
The solution is proposed for a problem that everybody agrees on. stakeholders collaborate with the development team to express and document their problems,

needs, and potential features for the system to be, so the project team can better understand what has to be done.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Analyst	Mandatory: • None	• #workproduct# Glossary • #workproduct# Vision
Additional: • #role# Architect • #role# Project Manager • #role# Stakeholder	Optional: • #workproduct# Vision	
Assisting:	External: • None	

.

Steps

.

- **Identify stakeholders:**

  Identify the stakeholders: decision-makers, customers, potential users, partners, domain experts, industry analysts and other interested parties. Briefly describe what stakeholders do and what their responsibilities are with regard to the system being developed.

- **Gain agreement on the problem to be solved:**

  Avoid rushing into defining the solution. First, gain agreement on the definition of the problem by asking the stakeholders what they see as the problem. Then search for root causes, or the "problem behind the problem". Use appropriate requirements gathering techniques. Formulate the problem statement. The purpose of this is to help you distinguish solutions and answers from problems and questions.

- **Gather stakeholder requests:**

  Use the most appropriate technique to help you on requirements gathering. Each technique is applicable in a particular situation or to a certain type of stakeholder.

  If you can meet stakeholders in person, then you can conduct an interview or a brainstorming session. This face-to-face collaboration is extremely valuable and reduces the chances of the project team misunderstanding the needs of the stakeholders.

  Some requirements might already be documented in other work products (such as in change requests or work items). This can often be used as a solid starting position from which a full set of requirements can be created.

- **Define the scope of the solution:**

  Analyze the scope in terms of processes, organizations, and systems.

- **Define features of the system:**

  Work with stakeholders to capture a list of features that stakeholders want in the system, briefly describing them and giving attributes to help define their general status and priority in the project.

- **Achieve concurrence:**

  Conduct an effective requirements review with stakeholders and the development team to ensure agreement on the project vision, assess quality, and identify required changes.

- **Capture a common vocabulary:**

  Every project has its own specialized terminology that everyone on the team must understand well to communicate effectively with stakeholders. Work with stakeholders to create a glossary that defines acronyms, abbreviations, and relevant business and technical terms. Work with stakeholders to continually expand and refine the glossary throughout the project lifecycle.

Key Considerations
Use-case modeling is one technique that can prove useful in defining the system boundaries and system behavior.

More Information

- **#checklist# Vision.**
- **#concept# Requirement Attributes.**
- **#guideline# Effective Requirement Reviews.**
- **#guideline# Requirements Gathering Techniques.**

.

.
2.2.1.2 #task# Plan Project

Summary
A collaborative task that outlines an initial agreement on how the project will achieve its goals. The resulting project plan provides a summary-level overview of the project.

Purpose
Get stakeholder buy-in for starting the project and team commitment to move forward with it. This plan can be updated as the project progresses based on feedback and changes in the environment.

Relationships.
.

Roles	Inputs	Outputs
Primary:	Mandatory:	• #workproduct# Project

• #role# Project Manager	• #workproduct# [Project Work] • #workproduct# [Technical Specification]	Plan
Additional: • #role# Analyst • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • None	
Assisting:	External: • None	

.

Main Description

Developing the project plan provides an opportunity for the team to agree on project scope, objectives, initial timeframe, and deliverables. It allows the team to begin demonstrating self-organization by defining success criteria and work practices to be used. Collaboration and consensus by all key project participants is the goal, but the role responsible for this task must ensure that everybody is committed to the plan.

Steps

- **Identify a cohesive team**:

  Revisit the resourcing for the project. Identify gaps and initiate hiring or re-allocation of resources as needed. Discuss with the team who plays which roles, and have them agree on their responsibilities.

- **Estimate project size**:

  The team produces rough size estimates for each work item found in the [Project Work].

  Discuss with stakeholders to determine what is realistic to develop within the constraints of the project. Use stakeholder priorities and estimates from the team to guide these discussions.

- **Forecast project velocity and duration**:

  Define the iteration length and use it to assess target velocity. The number of items to be delivered in each iteration will be set by the velocity of the team and the estimates for each item.

  If the project is feature-driven, the team uses the [Project Work] and target velocity to forecast the number of iterations required to complete the project. If the project instead is date-driven, the team assesses (using the known velocity of the team) roughly how much work can be done in the given time-frame. Out-of-scope work can be considered for future releases.

The team should not spend too much time doing this planning. The project plan should document only a brief summary of project milestones and between one and three objectives for each iteration. Do not commit individual work items to the plan, because this will force too much re-planning. The goal is just to create a high-level plan outlining how the team can build the resulting application in the given set of iterations. Extra levels of detail will be achieved in other planning sessions throughout the project (for example, when planning iterations). You may need to revisit this plan later to adapt it based on what you will learn by running the iterations.

- **Evaluate risks:**

  The team identifies project risks, performs a qualitative risk analysis to assess their order of magnitude, and updates the [Project Risk]. The project manager facilitates the team decision about which risks they should respond to, and which risks they should watch for.

  Responses may include avoiding or mitigating risks, exploring opportunities, or increasing the probability and positive impacts of the risk. Depending on the case, work items may have to be added or removed from the [Project Work] to make sure that responses will be prioritized and handled by the team along with other project work. Because it is not feasible to plan responses for all identified risks, the team may decide to accept some of them. Keep the risks to watch in the risks list, and communicate them to stakeholders. Determine actions only if they occur.

- **Establish costs and articulate value:**

  Develop a rough order of magnitude estimate for the costs of resources needed to complete project work items. A simplified project costing model can be applied by multiplying the approximate cost per person for the entire team by the length of an iteration to derive ongoing financial impact (that is, cost per iteration). This first round of planning should keep things very rough and flexible. The goal is just to articulate value against the budget constraints of the project, and to help stakeholders decide whether it is worth moving forward with the project or not. If necessary, propose options to decrease costs, such as removing low-value and high-cost work items from the scope .

- **Plan deployment:**

  Plan the strategy for deploying the software (and its updates) into the production environment as early as possible, because it may impact the [Project Work]. The team may need to discuss the release timeframe with the operations and support departments to ensure that the project fits into the overall corporate deployment system.

  Whenever possible, the team should consider deploying small releases (release cycles of three to four months at most). Releasing software into production frequently is a good way to get early feedback from the users, in order to enhance the overall product quality.

  Capture the objectives for deployment and release dates in the Project Plan.

- **Outline project lifecycle:**

Organize iterations into a set of phases. Each phase in the project lifecycle will end with a milestone aimed at providing stakeholders with oversight and steering mechanisms to control project funding, scope, risk exposure, value provided, and other aspects of the process (see **#concept# Project Lifecycle**).

More Information

- **#concept# Phase Milestones**
- **#guideline# [Project Planning Guidance]**
- **#guideline# Agile Estimation**
- **#guideline# Deploy the Solution**
- **#guideline# Managing Risks**
- **#guideline# Staffing a Project**

.

.

2.2.2 #activity# Plan and Manage Iteration

Summary

Initiate the iteration and allow team members to sign up for development tasks. Monitor and communicate project status to external stakeholders. Identify and handle exceptions and problems.

WBS

- #task# Plan Iteration
- #task# Prepare Environment
- #task# Manage Iteration
- #task# Assess Results

Description

This activity is performed throughout the project lifecycle. The goal of this activity is **to identify risks and issues early enough** that they can be mitigated, **to establish the goals** for the iteration, and **to support the development team** in reaching these goals.

The project manager and the team launch the iteration. The prioritization of work for a given iteration takes place. The project manager, stakeholders, and team members agree on what is supposed to be developed during that iteration.

Team members sign up for the **work items** they will develop in that iteration. Each team member breaks down the work items into development **tasks** and estimates the effort. This provides a more accurate estimate of the amount of time that will be spent, and of what can be realistically achieved, in a given iteration.

As the iteration runs, the team meets regularly to report status of work completed, the work to do next, and issues blocking the progress. In some projects, this status checking occurs in daily meetings, which allows for a more precise understanding of how the work in an iteration is progressing. As necessary, the team makes corrections to achieve what was planned. The overall idea is that risks and issues are identified and managed throughout the iteration, and everyone knows the project status in a timely manner.

During iteration assessments, the key success criterion is the demonstration that planned functionality has been implemented. Lessons learned are captured in order to modify the project or improve the process. If the iteration end coincides with

the phase end, make sure the objectives for that phase have been met (see **#concept# Phase Milestones for more information**).

.

.

2.2.2.1 #task# Plan Iteration

Summary
Plan the scope and responsibilities for a single iteration.

Purpose
The purpose of this task is to identify the next increment of system capability, and create a fine-grained plan for achieving that capability within a single iteration.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Project Manager	Mandatory: • #workproduct# Work Items List	• #workproduct# Iteration Plan • #workproduct# Risk List • #workproduct# Work Items List
Additional: • #role# Analyst • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • #workproduct# [Technical Architecture] • #workproduct# [Technical Specification] • #workproduct# Iteration Plan • #workproduct# Risk List	
Assisting:	External: • None	

.

Main Description

.

During project planning, iterations are identified, but the estimates have an acceptable uncertainty due to the lack of detail at the project inception. This task is repeated for each iteration within a release. It allows the team to increase the accuracy of the estimates for one iteration, as more detail is known along the project.

Ensure that the team commits to a reasonable amount of work for the iteration, based on team performance from previous iterations.

Steps

- **Prioritize Work Items List**

Prioritize the work items list before you plan the next iteration. Consider what has changed since the last iteration plan (such as new change requests, shifting priorities of your stakeholders, or new risks that have been encountered).

- **Define iteration objectives**

  Work with the team to refine the iteration objectives found in the project definition and scope, and document them in the iteration plan in order to provide high-level direction to what should be targeted for the iteration. The objectives should be driven based on stakeholder priorities, and will be revised as the iteration plan is finalized. Those objectives are usually defined as high-level capabilities or scenarios, which need to be implemented and tested during the iteration in order to deliver increased value to the customer.

- **Identify and review risks**

  Throughout the project, new assumptions and concerns may arise. Help the team identify and prioritize new risks as part of iteration planning, updating the risk list. Add risk responses to the work items list, influencing the work that is being planned for that iteration.

- **Commit work to the iteration**

  Work with the team, and especially the project stakeholders, to identify the high-priority work items from the work items list to be addressed. The high-level objectives provide guidance on what work items should be considered. The iteration plan from previous iteration should include an assessment of the results, and can also be used as input to the current iteration planning. The team reviews its velocity and determines the amount of work that can be done within the iteration. The team breaks down into tasks those work items that are assigned to the iteration, and estimates the effort to complete each task. Typical tasks range from half a day to two days in length, and are captured in the work items list. See Guideline: Agile Estimation for more information.

  When the team has decided to take on a work item, it will assign the work to one or several team members. Ideally, this is done by team members signing up to do the work, since this makes people motivated and committed to doing the job. However, based on your culture, you may instead assign the work to team members.

- **Define evaluation criteria**

  Each iteration should include testing as a part of the evaluation, as well as the test objectives and test cases that need to be detailed. Other evaluation criteria may include successful demonstrations to key stakeholders, or favorable usage by a small group of target users. Document evaluation criteria in the iteration plan.

- **Refine project definition and scope**

  Depending on the results of the previous iteration assessment, update the project definition work products as needed. Necessary changes can encompass the need to acquire new resources, to absorb an unplanned effort increase, or to

implement a specific change request. If a change affects defined project milestones, consult with the stakeholders before committing to it.

More Information

- **#guideline# [Project Planning Guidance]**
- **#guideline# Agile Estimation**
- **#guideline# Assign Changes to an Iteration**
- **#guideline# Deploy the Solution**
- **#guideline# Iteration Planning**
- **#guideline# Managing Risks**
- **#guideline# Prioritizing Work Items**

.

2.2.2.2 #task# Manage Iteration

Summary

Assess project status and identify any blocking issues and opportunities. Identify and manage exceptions, problems, and risks. Communicate project status.

Purpose

Help the team meet the iteration objectives and keep the project on track. Manage stakeholders' expectations as technical and practical discoveries are made during the project.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Project Manager	Mandatory: • #workproduct# [Project Definition and Scope] • #workproduct# Iteration Plan • #workproduct# Risk List • #workproduct# Work Items List	• #workproduct# Iteration Plan • #workproduct# Risk List • #workproduct# Work Items List
Additional: • #role# Analyst • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • None	
Assisting:	External: • None	

.

Main Description

Developing the team is part of executing the project. Improve the interactions and trust between team members, incorporating team building into project activities,

and empowering the team to make decisions. Management is done by objectives and not by time reporting.

Help the team meet the iteration objectives by removing constraints, and by monitoring the progress and work remaining to completion. When the team is falling behind, help the team assess how it can reduce work and still meet the iteration goals. Involve stakeholders in approving changes and actions that affect them, so that their objectives for the iteration can be met, and the issues they raise can be solved throughout the iteration.

Select an approach for the collection of basic metrics: ideally, they are automatically generated from the tools at hand, but can be manually assembled, or both. Outline which metrics the project should use in the [Project Definition and Scope].

Steps

- **Track current iteration progress**

  Continuously monitor the iteration to ensure that it is progressing appropriately. Track the progress of the current iteration by maintaining visibility on the status of the items that are being worked on during the iteration. In particular, it is important to be able to understand how quickly the team is moving through the work scheduled for the iteration, and how accurate the estimates were [LEF07].

  Share information about the progress being made by having quick, frequent meetings with the entire project team. These meetings are useful to understand what team members have accomplished since the last meeting, and what they plan to accomplish before the next meeting. It also allows the team to identify any blocking issues. See [Collaboration Guidance] for more information.

  Track the status of each work item by using visual indicators of state information (for example: defined, completed, accepted) in a wall or automated tool. Since iterations are typically fixed in duration, another primary way to gauge progress is to continuously monitor current status and also estimate how much work remains. Plot an iteration burndown chart on a daily basis in order to visualize progress on a given iteration. This chart is plotted with the total estimates for all the items that are not yet started and the estimated remaining effort for any item in progress. See Report: Iteration Burndown for more information.

  Take corrective actions to adjust the course of the iteration if the burndown chart shows that the work will not be finished as planned.

- **Capture and communicate project status**

  Because outside stakeholders do not participate in the daily activities of the team, it is important that the status of the project should be communicated to them as often as possible. This communication significantly lowers the risk of disconnect between the development team and the stakeholders. The status provides the team with data they can use to improve their development process.

  Besides the iteration status, the overall project status can be captured and communicated in a burndown report for the whole project. This report should show the progress toward the release date, and should be updated at the end of every iteration (when metrics, such as the team's velocity, can also be provided). This allows the team to predict what it can accomplish on the

remaining project iterations. Other metrics (such as number of test cases passed) can also be used to present iteration and project status.

Keep the status information visible to stakeholders and the project team at all times in a project workspace (walls or automated tool), where stakeholders can come and experience first-hand the progress being made by the team.

- **Handle exceptions and problems**

  Know about the team's problems and issues and focus on problems that are blocking progress. A quick, daily meeting is usually a good way to monitor those problems and issues. Keep a record of issues that have to be solved within the team in the Iteration Plan. If a critical problem or issue is likely to take a lot of time to solve, capture it in the Work Items List so that it can be assigned to an individual.

  Identify the cause and impact of problems and exceptions as they arise. Identify possible solutions for problems that have an immediate impact on the short-term goals and objectives. Identify who needs to be involved in implementing the solution. Define the corrective actions and implement them.

- **Identify and manage risks**

  Identify risks as soon as the project starts, and continue identifying and managing risks throughout the project. Revisit the Risk List weekly, or at a minimum once per iteration. Involve the entire team in identifying and mitigating risks.

- **Manage objectives**

  When a team is falling significantly behind, or critical problems occur that prevent the team from meeting the iteration objectives, it may be necessary to descope work to ensure that the team delivers a useful product increment by the end of the iteration, while maximizing stakeholder value. Work with the team and stakeholders to revise the Iteration Plan and, as necessary, reduce the emphasis on less critical tasks by postponing them to a subsequent iteration. In rare cases, if the iteration objectives still seem impossible to meet, the team might consider terminating the iteration or reformulating the iteration to a new objective.

.

More Information

- **#concept# Metrics**
- **#concept# Risk**
- **#guideline# [Collaboration Guidance]**
- **#guideline# Assign Changes to an Iteration**

.

2.2.2.3 #task# Assess Results

Summary

Determine success or failure of the iteration. Apply the lessons learned to modify the project or improve the process.

Purpose

Demonstrate the value of the solution increment that was built during the iteration and apply the lessons learned to modify the project or improve the process.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Project Manager	Mandatory: • #workproduct# Iteration Plan • #workproduct# Work Items List	• #workproduct# Iteration Plan • #workproduct# Work Items List
Additional: • #role# Analyst • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • #workproduct# [Project Definition and Scope] • #workproduct# [Technical Specification] • #workproduct# [Technical Test Results]	
Assisting:	External: • None	

.

Main Description

Coordinate the assessment and discuss with the team how the iteration results will be best presented to stakeholders, so that they can learn as much about the solution as possible. Listen to what the team has to say about what went wrong (and what went right) during the iteration. This knowledge will help everybody make informed decisions about the next iteration planning, and determine the best course of action for the project. This task is performed at the end of every iteration until the end of the project.

Steps

- **Prepare for iteration assessment**

  Towards the end of the iteration, the team jointly assesses whether the objectives and evaluation criteria established in the Iteration Plan were met, and whether the team adhered to the plan and completed all of the work items committed to the iteration. The team makes use of objective measures to the greatest extent possible. To assess that a given work item is completed, the team ensures that the corresponding test cases were successfully run against it.

  The team prepares a demonstration of the features implemented at that point, so that during the iteration assessment stakeholders can have a real sense of progress made. The team decides whether each developer should demonstrate the features that they implemented, or if the project manager or senior developer demonstrates it all, with other team members present to answer questions.

  In addition to the demonstration, prepare reports that show project status, such as work burndown and test case reports.

These activities happen in preparation for the iteration assessment meeting with stakeholders that occurs on the last day of the iteration.

- **Demonstrate value and gather feedback**

  The team demonstrates the product to customers, end-users, and other stakeholders to collect their feedback or, better yet, have end users use the product themselves. This can be done throughout the iteration, but at least during the iteration assessment that occurs at the end of the iteration (see Guideline: Iteration Assessment). Work that is not completed should not be demonstrated.

  Record resulting knowledge (such as new functionality, requested changes, and defects) in the Work Items List, so that project priorities, scope, and duration can be refined in the next iteration planning.

- **Perform a retrospective**

  Review with the team the approach taken to development and collaboration, the effectiveness of the development environment, the suitability of the working environment, and other factors. Discuss what things went well, what could have gone better, and how things could be changed to deliver better results. Capture in the current Iteration Plan the assessment results, stakeholder feedback, and actions to be taken to improve the development approach for the next iteration. Record lessons learned in this iteration with a collection of lessons learned for the entire project.

- **Close-out project**

  Perform this step when the iteration review coincides with the end of the project. Involve the team and stakeholders in a final assessment for project acceptance which, if successful, marks the point when the customer accepts ownership of the software product. Complete the close-out of the project by disposing of the remaining assets and reassigning the remaining staff.

- **Perform a retrospective (end of phase)**

  When the assessment period coincides with the end of a phase, the corresponding milestone review takes place. These are informal reviews of the work accomplished where the team and stakeholders agree on moving the project on to the next phase, spanning a set of iterations with a new common goal in accordance with the emphasis of the following phase. For more information, see **#concept# Phase Milestones.**

More Information

- **#concept# Phase Milestones**
- **#concept# Retrospective**
- **#guideline# Conduct Project Retrospective**
- **#guideline# Deploy the Solution**
- **#guideline# Iteration Assessment**
- **#toolmentor# Use Method Composer to Update the Process**

.

.

2.2.3 #activity# Prepare Environment

2.2.3.1 #task# Tailor the Process

Summary
This task describes how to tailor a process for a project.

Purpose
The purpose of this task is to ensure that the project team has a defined process that meets their needs.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Project Manager	Mandatory: • #workproduct# Project Defined Process	• #workproduct# Project Defined Process
Additional: •	Optional: • None	
Assisting: •	External: • None	

.
Steps
.

- **Define the scope of the tailoring effort**

  Determine what processes and tools are mandated by the organization. Determine which process areas to document or tailor. You may choose to divide this task into separate tasks for each process area. See guidelines for typical process areas.

- **Involve appropriate stakeholders**

  Depending on the process areas you intend to tailor, you will want to involve the affected team members. For example, changes to coding guidelines should involve coders, changes to governance guidelines should involve stakeholders that govern the project, project acceptance procedures should involve the customer, etc.

- **Decide how the process will be documented**

  You can use Rational Method Composer configurations, process documents such as a Development Case, RACI tables, and so on. See **#guideline# on documenting the project process.**

- **Develop project-specific content**

Identify reusable processes/process assets including applicable practices, guidelines, and examples.

Decide which tools will be used to create and maintain work products

- **Review and approve the process**

.

.

2.2.3.2 #task# Set Up Tools

Purpose
The purpose of this task is to:

- Install the tools
- Customize the tools.
- Make the tools available to the end users.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Tool Specialist	Mandatory: • #workproduct# Tools	• #workproduct# Tools
Additional: •	Optional: • None	
Assisting: •	External: • None	

Main Description
Many software-development tools support teams of people, with several users working with information stored in a shared repository. Each user uses the tool on their personal computer (client) and the shared repository is stored on a central server. In this case the tool must be installed on the server and on the clients. Customizing the tool is done both on the server and on the client.

There are tools that do not use a shared repository, such as compilers, debuggers, editors, graphic tools, etc. These tools can simply be installed on the users' computers. It may still be needed to customize the tools so that all members of the project use the tool in the same way.

The approach should be to automate as much as possible of the installation and customization procedures.

.
Steps
.

- **Install the Tool on the Server**

Identify what other software is required for the specific tool to work, and install this software. For example, a tool may require a database management system (DBMS) be installed first.

When you have installed the support software, you can install the tool on the server.

- **Customize the Tool**

  Decide how to customize the tool so that it supports the Project Defined Process in the best way.

  In addition to customizing the tools, you should set up user groups and access rights on the server. In some cases, a tool may provide its own mechanisms for this. In other cases, user groups and access rights are defined using the operating system. The configuration of user groups and access rights affects how the tools can be used. For example, you can set constraints on what parts of a repository certain users will have access to.

  Document the customizations in project-specific guidelines.

- **Integrate with Other Tools**

  Integrate the tool with other tools to make it easier to use. An integration between tools is in most cases in the form of an extension to one or several tools. An 'integration extension' to a tool typically:

  - Synchronize data between the different tools. It automates the creation and maintenance of related items in development projects
  - Automatically adds traceability between related items in different tools.
  - Allow the user to add traceability between items in different tools.
  - Allow the user to navigate between tools. For example, access an item in a test tool from a requirements management tool.
  - Allow the user to run certain functionality from one tool. For example, the possibility to create items in another tool.
  - Allow the user to version a tool's items in a configuration management tool.

  Most tools offer ready-to-use extensions to integrate tools with each other.

  Describe how the tools are integrated with each other in the project-specific guidelines.

- **Install and Customize Tools on Clients**

  Install the tool on each client. The least that is needed to do when installing a tool on the client side, is to set up the connection to the repository on the server.

  Customize the tool on the clients, just as you customized the tool on the server:

  - In some cases you do not have to do anything with the client. For example, if the client is a web-interface it is enough that the clients get the address to the application on the server. Some tools allow you to do all customization on the server side. When the users access the repository on the server, they automatically get the correct settings.

    - In other cases you customize the tool on the client by installing software that customizes the tool, or installing files with customization information.

  It may be necessary to install 'integration software' on the client. Place the 'integration software' on a server and allow the users to download and install it on their computers.

  If it possible you should automate the tool installation, and the tool customization for the users. The benefit of creating installation programs is that it allows you to set up the tools so that the clients get all the right settings, extensions, and connections to the repository. You create installation (and customization) programs, and place them on a server. Then the users download these programs and run them to install and customize the tool in their computer.

.

.

2.2.3.3 #task# Verify Tool Configuration and Installation

Summary
This task describes how to verify that the Development Configuration is ready to be used by the project.

Purpose
The purpose of this task is to verify that the tools can be used to develop the system.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Tool Specialist	Mandatory: • #workproduct# Tools	• #workproduct# Tools
Additional: •	Optional: • None	
Assisting: •	External: • None	

Main Description
The tools and the development infrastructure has to be verified before the project starts using them. How this is done will clearly vary dependent upon skills, technology, and tools.

Steps
.

- **Verify the Environment**

  Verify the environment contains the correct hardware, software, and data. Verify that the correct hardware is installed. This may be done visually or

through a tool (such as using Windows Properties for My Computer).

- **Verify the Tools**

  Verify that the correct software configuration is installed. This may be done by looking at the registry settings, 'ini' files, or by launching the tool and looking at some information options or configuration options.

- **Verify Data**

  Verify the data contains the appropriate data. Verifying data may require using tools to visually inspect the data, or using an application to display the data. At some point, one or more use cases may be selected and executed (one or more scenarios) for each tool to ensure the tool and the results of using the tool are consistent with the need.

- **Run the Tools**

  Assemble a small team of people who know the Toolsand the Project Defined Process well, and let them run the tools.

  - Test multi site, many colliding users
  - At least one use case scenario for each tool has been executed to verify the appropriate installation and configuration of the tools.
  - Try the normal scenarios of the development process and in the tool guidelines.
  - Test the integration between different tools.

  Issue change requests if necessary.

.

.

2.2.3.4 #task# Deploy the Process

Summary
This task describes how to rollout a development process to the project team.

Purpose
The purpose of this task is to:

- Ensure that the project members are properly introduced to the process.
- Harvest any feedback on the process and refine the process, as necessary.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Process Engineer	Mandatory: • #workproduct# Project Defined Process	• #workproduct# Project Defined Process
Additional: •	Optional: • None	

Assisting: •	External: • None	

Steps

.

- **Make the changes public**

  Inform all project members about any significant updates.

- **Educate project members**

  Unless the change is trivial, you need to educate the project members about the new Development Process, including any guidelines, templates and/or tools. This can vary from an informal 2 hour presentation to more formal training, depending on the size of the project and the project members' familiarity with similar development processes.

  The following are commonly used ways to educate the project members:

  - Briefing

    If the change is small or easy to understand a briefing to the team may be sufficient.

  - Mentors

    Having experienced mentors or consultants on the project or available to the project is an effective way to provide guidance on an as-needed basis.

  - Workshop

    A one day process adoption workshop (PAW) for all project members can help "kick-start" a new or significantly changed process. See Guidelines: Process Adoption Workshop

  - Customized training courses.

    If the project member have not attended the standard training courses in process and tools, an alternative is to customize the standard training courses, to cover the project's development process, including guidelines, templates and tools. However, it can be expensive to customize training courses. Generic process training, like an introductory course to the Process or one or more practices, should be conducted prior to project startup, or in the early days of the project. More specialized training in techniques, methods or technologies, is often conducted "just-in-time". This means that the training is given shortly before the method or technique is to be applied in the project, to ensure that new knowledge is fresh in mind. "Boot-camps".

  - 1-5 weeks of concentrated hands-on training. Not many organizations can afford to arrange these kinds of boot-camps, but they have proven to be efficient if there are many new factors for the people in the project. A boot-camp is typically a mixture of seminars, training courses and hands-on work with the process and tools.

- **Collect feedback**

More Information

- **#guideline# Deploying Tools**

.

.

2.2.4 #activity# Identify and Refine Requirements

Summary
Detail a set of requirements (one or more use cases, scenarios or system-wide requirements).

Description
This activity describes the tasks you perform to gather, specify, analyze, and validate a subset of system's requirements prior to implementation and verification. This does not imply that all requirements are detailed prior to commencing implementation. Rather, you perform this activity throughout the lifecycle with stakeholders and the entire development team collaborating to ensure that a clear, consistent, correct, verifiable, and feasible set of requirements is available, as needed, to drive implementation and verification.

During **Inception**, the focus is on gaining agreement on the problem to be solved, gathering stakeholder needs, and capturing high-level system features.

During **Elaboration**, the focus shifts to defining the solution. This consists of finding those requirements that have the most value to stakeholders, that are particularly challenging or risky, or that are architecturally significant. You then describe requirements (that are prioritized, via the work items list, for implementation in the early iterations) in sufficient detail to validate the development team's understanding of the requirements, to ensure concurrence with stakeholders, and to permit software development to begin. For each of these requirements, define associated test cases to ensure that the requirements are verifiable, and to provide the guidance needed for verification and validation.

During **Construction**, the focus shifts to refining the system definition. This consists of detailing the remaining requirements and associated test cases as necessary to drive implementation and verification, and managing requirements change.

.

.

2.2.4.1 #task# Identify and Outline Requirements

Summary
This task describes how to identify and outline the requirements for the system so that the scope of work can be determined.

Purpose
The purpose of this task is to identify and capture functional and non-functional requirements for the system. These requirements form the basis of communication and agreement between the stakeholders and the development team on what the system must do to satisfy stakeholder needs. The goal is to understand the requirements at a high-level so that the initial scope of work can be determined. Further analysis will be performed to detail these requirements prior to implementation.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Analyst	Mandatory: • None	• #workproduct# Glossary • #workproduct# System-Wide Requirements • #workproduct# Use Case • #workproduct# Use-Case Model • #workproduct# Work Items List
Additional: • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • #workproduct# [Technical Specification]	
Assisting: •	External: • None	

Steps

.

- **Gather information**

  Use various techniques to make gathering requirements easier. Face-to-face meetings with stakeholders is the most effective way to understand stakeholder needs and to gather and validate requirements, but you must prepare in order for these meetings to run efficiently.

  Be prepared by gathering and reviewing information related to the problem domain, problem statement, business environment and key stakeholders. Most of this information must be available in the vision. Also review the existing requirements repository for stakeholder requests.

- **Identify and capture domain terms**

  If there are ambiguous or domain-specific terms that need to be clearly defined, make sure you work closely with stakeholders to capture these terms in the glossary and that you use these terms consistently.

- **Identify the types of requirements relevant to your system**

  Requirements can be broadly classified as either functional or non-functional requirements. The former specify what the system must do. The latter specify constraints on the solution such as usability, reliability, performance, supportability, interfaces with legacy systems, etc. Depending upon the domain there might be regulatory requirements that apply.

  Collaborate with stakeholders to identify the types of requirements relevant to your system. This will help you assess the completeness of your requirement set.

- **Identify and capture use cases and scenarios**

  Collaborate with stakeholders to identify and capture the use cases and scenarios relevant to your system. Capture references to these requirements with other project to-do items in the #workproduct# Work Items List so that you can prioritize the work.

  See **#guideline# Identify and Outline Actors and Use Cases** for more information.

- **Identify and capture system-wide requirements**

  Collaborate with stakeholders to identify and capture the system-wide requirements relevant to your system. Capture references to these requirements with other project to-do items in the #workproduct# Work Items List so that you can prioritize the work.

  See **#guideline# Developing System-Wide Requirements Specification** for more information.

- **Achieve concurrence**

  Conduct a review of the requirements with relevant stakeholders and the development team to ensure consistency with the agreed vision, assess quality, and identify any required changes.

- **Identify and capture Use Cases and Actors in a Use-Case Model**

  Find and define the line that divides the solution and the real world that surrounds the solution. Collaborate with the project manager and architect, because decisions concerning system boundaries will have a major impact on cost, schedule and system architecture.

  Collaborate with stakeholders to identify interfaces, as well as input and output information exchanged with users, machines, or systems. Identify and capture the **#concept# Actors** and #workproduct# Use Cases in the #workproduct# Use-Case Model. See **#guideline# Identify and Outline Actors and Use Cases** for more information.

More Information

- **#concept# Requirements**
- **#concept# System-Wide Requirements**
- **#guideline# Developing System-Wide Requirements Specification**
- **#guideline# Effective Requirement Reviews**
- **#guideline# Identify and Outline Actors and Use Cases**
- **#guideline# Requirements Gathering Techniques**

.

.

2.2.4.2 #task# Detail Use-Case Scenarios

Summary
This task describes how to detail a use-case scenarios for the system.

Purpose
The purpose of this task is to describe use-case scenarios in sufficient detail to validate understanding of the requirements, to ensure concurrence with stakeholder expectations, and to permit software development to begin.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Analyst	Mandatory: • #workproduct# Use Case	• #workproduct# Glossary • #workproduct# Use Case • #workproduct# Use-Case Model
Additional: • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • None	
Assisting: •	External: • None	

Steps
.

- **Detail use cases and scenarios**

  Some use cases and scenarios might need to be described in more detail to validate the understanding of the requirements and to permit software development to begin. This does not imply that all use cases and scenarios will be detailed prior to commencing implementation on them. The level of detail captured will vary depending upon the needs of the project and the complexity of the use case. Capture the use case and scenarios details in the use case specification.

- **Detail glossary terms**

  Review the flow of the use case or scenario. If information is exchanged, be specific about what is passed back and forth. Collaborate with stakeholders to ensure that you define newly discovered domain terms, or ambiguous terms properly in the glossary. If your understanding of the domain has improved, refine existing glossary terms.

- **Achieve concurrence**

  Review the detailed use case and scenarios with relevant stakeholders and the development team to ensure consistency with the agreed vision. Assess quality and identify any required changes.

- **Update Use-Case Model**

Based on your work detailing use cases and scenarios, update the #workproduct# Use-Case Model. Add, remove or update **#concept# Actors** and #workproduct# Use Cases as required.

For more information on creating and structuring your use case model see **#guideline# Use-Case Model**. For assistance in assessing the quality of your use-case model see **#checklist# Use-Case Model**.

Key Considerations
To avoid unnecessary rework, only those use-case scenarios that are **scheduled for implementation in the near term** (in the next iteration or two) must be detailed. Not all use-case scenarios require detailing.

More Information

- **#guideline# Detail Use Cases and Scenarios**
- **#guideline# Effective Requirement Reviews**

.

.
2.2.4.3 #task# Detail System-Wide Requirements

Summary
This task details one or more requirement that does not apply to a specific use case.

Purpose
The purpose of this task is to describe one or more system-wide requirements in sufficient detail to validate understanding of the requirements, to ensure concurrence with stakeholder expectations, and to permit software development to begin.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Analyst	Mandatory: • #workproduct# System-Wide Requirements	• #workproduct# Glossary • #workproduct# System-Wide Requirements
Additional: • #role# Architect • #role# Developer • #role# Stakeholder • #role# Tester	Optional: • #workproduct# Use Case	
Assisting: •	External: • None	

Steps
.

- **Detail system-wide requirements**

Some system-wide requirements might need to be clarified or described in more detail. New requirements may be discovered while the use cases and scenarios are detailed, and new requirements may have been submitted as change requests. Collaborate with stakeholders to capture, refine and validate those requirements that will have an impact on near term, planned work or are deemed architecturally significant.

- **Detail glossary terms**

  Review the system-wide requirements and collaborate with stakeholders to ensure that you define newly discovered domain terms, or ambiguous terms properly in the glossary. If your understanding of the domain has improved, refine existing glossary terms.

- **Achieve concurrence**

  Review the detailed system-wide requirements with relevant stakeholders and the development team to ensure consistency with the agreed vision. Assess quality and identify any required changes.

Key Considerations
To avoid unnecessary rework, only those requirements that are **scheduled for implementation** in the near term (in the next iteration or two) must be detailed.

More Information

- **#guideline# Developing System-Wide Requirements Specification**
- **#guideline# Effective Requirement Reviews**

.

.
2.2.4.4 #task# Create Test Cases

Summary
Develop the test cases and test data for the requirements to be tested.

Purpose
To achieve a shared understanding of the specific conditions that the solution must meet.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Tester	Mandatory: • #workproduct# [Technical Specification]	• #workproduct# Test Case
Additional: • #role# Analyst • #role# Developer • #role# Stakeholder	Optional: • #workproduct# Test Case	
Assisting:	External:	

•	• None	

Steps

.

- **Review the requirements to be tested**

  Work with analysts and developers to identify which scenarios and requirements need new or additional test cases. Review the Plans to ensure you understand the scope of development for the current iteration.

- **Identify relevant Test Cases**

  Identify paths through the scenario as unique test conditions. Consider alternative or exception paths from both a positive and negative perspective. Review the test ideas list for patterns of test cases that apply to the scenario.

  Discuss the requirement with stakeholders to identify other conditions of satisfaction for the requirement.

  List the test cases with a unique name that identifies the condition they evaluate or their expected result.

- **Outline the Test Cases**

  For each test case, write a brief description with an expected result. Ensure that a casual reader can clearly understand the difference between test cases. Note the logical pre-conditions and post-conditions that apply to each test case. Optionally, outline steps for the test case.

  Verify that test cases meet the **#checklist# Test Case guidelines**.

- **Identify test data needs**

  Review each test case and note where data input or output might be required. Identify the type, quantity, and uniqueness of the required data, and add these observations to the test case. Focus on articulating the data needed and not on creating specific data.

  For more information on test data selection, see **#checklist# Test Data**.

- **Share and evaluate the Test Cases**

  Walk through the test cases with the analysts and developers responsible for the related scenario. Ideally, the stakeholders will also participate.

  Ask the participants to agree that if these test cases pass, they will consider these requirements implemented. Elicit additional test ideas from Analysts and stakeholders to ensure you understand the expected behavior of the scenario.

  During the walkthrough, ensure that:

  - The requirements planned for the current iteration have test cases.
  - All the participants agree with the expected results of the test cases.

    - There are no other conditions of satisfaction for the requirement being tested, which indicates either a missing test case or a missing requirement.

Optionally, capture new patterns of test cases in the test ideas list (see **#concept# Test Ideas**).

Key Consideration
Develop test cases in parallel with requirements so that Analysts and stakeholders can agree with the specific conditions of satisfaction for each requirement. The test cases act as acceptance criteria by expanding on the intent of the system through actual scenarios of use. This allows team members to measure progress in terms of passing test cases.

More Information

- **#guideline# Test Ideas**

.

.
2.2.5 #activity# Agree on Technical Approach

Summary
Reach agreement on a viable technical approach to developing the solution.

Description
The goal of this activity is to define a technical approach to the system that supports the project requirements, within the constraints placed on the system and the development team. The architect should do the following:

- Work with the team to create an initial sketch of the **technical approach** to the proposed system
- Ensure that the technical decisions are adequately captured and communicated
- Ensure that the team has enough information to understand the approach that your are taking

The work done here does not seek to produce a detailed and comprehensive technical specification for the system. Rather, the approach should be to decide the overall technical approach at a high level.

You should focus on proving the architecture with working software. If the solution is similar to a previously produced solution (or is a well-known solution domain), then it will probably be good enough to reference that example as evidence of the feasibility of the approach. In some cases, it may be necessary to develop one or more prototypes to validate some of the decisions or clarify some of the requirements.

The conclusion of this work should produce just enough information to communicate the architecture to the team, and to demonstrate its viability to the customer. This allows the project to move forward, enabling you to refine and baseline the architecture.
.

.
2.2.5.1 #task# Envision the Architecture

Summary
This task is where the "vision" for the architecture is developed through

- **analysis of the architecturally significant requirements**
- **identification of architectural constraints, decisions and objectives.**

Purpose
To envision a technical approach to the system that supports the project requirements, within the constraints placed on the system and the development team. To provide sufficient guidance and direction for the team to begin development.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Architect	Mandatory: • #workproduct# [Technical Specification]	• #workproduct# Architecture Notebook
Additional: • #role# Analyst • #role# Developer • #role# Project Manager • #role# Stakeholder	Optional: • #workproduct# Architecture Notebook	
Assisting: •	External: • None	

.
Main Description
This task focuses on envisioning the initial architecture and outlining the architectural decisions that will guide development and testing. It relies on gathering experience gained in similar systems or problem domains to constrain and focus the architecture so that effort is not wasted in re-inventing architecture.

The results are captured for future reference and are communicated across the team. It is important that the team has enough information to understand the technical approach being taken.

The architecture evolves organically over time by outlining and refining portions of it. A few people get together in a room and sketch out what they think the architecture will be. This envisioning effort sets the foundation for prototyping. If the solution is similar to a previously produced solution (or is a well-known solution domain), then it will probably be good enough to reference that example as evidence of the feasibility of the approach. In some cases, it may be necessary to develop one or more prototypes to validate some of the decisions or clarify some of the requirements.

The work done here does not seek to produce a detailed and comprehensive technical specification for the system. Rather, the approach should be to decide the overall technical approach at a high level. The conclusion of this work

should produce just enough information to communicate the architecture to the team, and to demonstrate its viability to the customer. This allows the project to move forward, enabling you to refine and baseline the architecture.

.

Steps

.

- **Identify architectural goals**

  Work with the team to describe the remaining goals for the architecture and identify which ones are appropriate to address for this iteration. Look at the requirements and speak to the people asking for them to make sure that the critical goals for this iteration are well understood. These goals will prioritize and guide the approach to important technical decisions.

  It's important to regularly review the status of these goals throughout the project to make sure that they are still valid and that the system is on track to deliver them.

  For more information, see **#concept# Architectural Goals.**

- **Identify architecturally significant requirements**

  Identify which of the current requirements are architecturally significant. Explore and refine those that must be implemented in order to realize the architectural goals for the current iteration.

  See **#concept# Architecturally Significant Requirements** for more information.

- **Identify constraints on the architecture**

  List any constraints on the architecture and any trade-offs between competing requirements and resources. Decide how the architecture will meet these issues. Justify each of the decisions made and capture this information. Regularly review the list of constraints to make sure that they are still valid and that no new ones have appeared.

  For more information, see **#concept# Architectural Constraints.**

- **Identify key abstractions**

  Identify the key concepts and abstractions that the system needs to handle. The requirements are good sources for key abstractions. Don't spend too much time describing abstractions in detail at this initial stage, because there is a risk that spending too much time will result in identifying classes and relationships that the solution does not actually need.

  When identifying key abstractions, it can be useful to also define any obvious relationships that exist between them. These can be captured in a table or in diagrams (in a tool or whiteboard. In general, it is not worth agonizing over defining a highly detailed set of relationships at this early stage in design. The relationships will become more concrete and detailed later and will probably modify these early assumptions.

  For more information, see **#concept# Key Abstractions.**

- **Identify reuse opportunities**

Survey, assess, and select available assets. Identify assets from other areas that may be reused in the current architecture.

For more information, see **#guideline# Software Reuse**.

- **Define approach for partitioning the system**

  Decide how to partition the software, both in logical and physical terms. Partitioning your system helps you manage its complexity by using the well-known "divide and conquer" strategy. By breaking the process into smaller and more manageable pieces, you make development easier. As a minimum, decide on:

  - How to partition the software when managing development (the use of layering as a partitioning strategy, for example). For more information, see **#guideline# Layering**.
  - How the software will be composed at run time.

  For each software partition, briefly describe

  - Its name and purpose.
  - Its relationships to other partitions.

  At this point, you do not need to identify the elements that should be placed in each of these partitions. Instead, you define how many partitions you will need and how they should be related. Later, during design activities, you decide which elements will populate these partitions.

- **Define approach for deploying the system**

  Outline how the software will deploy over the nodes on the network. Work with stakeholders such as network support and deployment teams to ensure that the proposed approach is a good fit for the wider technical environment.

- **Identify architectural mechanisms**

  Make a list of the technical services that the system needs to provide and capture some basic information about each item on the list. It's generally a good idea to make an initial list of all the mechanisms required for the project and then prioritize the development of those that need to be delivered to achieve the architectural goals.

  At this point, usually only the analysis mechanisms are defined. However, specific **#concept# Architectural Constraints** may mean that some of those mechanisms can be described as design mechanisms (even at this early stage).

  For more information on architectural mechanisms, see **#concept# Architectural Mechanism**.

- **Identify interfaces to external systems**

  At this point, identify the external systems with which this system must interact. An external system may be anything from software to hardware units that the current system will use, such as printers, terminals, alarm devices, and sensors.

Describe those interfaces at a high level, concentrating on the information that must pass between the systems.

- **Verify architectural consistency**

  Work with the team to verify that the architecture is consistent with the requirements and that the descriptions of the architecture are complete, meaningful, and clear.

- **Capture and communicate architectural decisions**

  Capture important decisions about the architecture in the Architecture Notebook for future reference. Make sure that the team understands the architecture and can deliver it.

.

Key Considerations

It is important to reduce the complexity of the solution by raising the levels of abstraction. For more information, see **#guideline# Abstract Away Complexity.**

It is critical that this task be performed collaboratively with active involvement of other team members and project stakeholders so that consensus and common understanding is reached. It is particularly vital to involve the developer(s) throughout this task. The architecture effort is about providing leadership and coordination of the technical work rather than putting in a solo performance.

At this stage, you may find it useful to develop a draft version of your architectural models. For more information, see **#guideline# Modeling the Architecture.**

.

Alternatives

This task is most needed when developing new and unprecedented systems. In systems where there is already a well-defined architecture, this task may be omitted and replaced with a review of the existing architecture.

More Information

- **#checklist# .**
- **#concept# Architectural Constraints.**
- **#concept# Architectural Goals.**
- **#concept# Architecturally Significant Requirements.**
- **#concept# Architectural Mechanism.**
- **#concept# Key Abstractions.**
- **#example#**
- **#guideline# Abstract Away Complexity.**
- **#guideline# Layering.**
- **#guideline# Modeling the Architecture.**
- **#guideline# Software Reuse.**
- **#template# .**

.

.