2.5 #phase# Transition Phase

- #iteration# Transition Iteration [1..n]
- 2.5.1 #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
- 2.5.2 #activity# Prepare Environment (same as 2.2.3)
  - 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
- 2.5.3 #activity# Develop Solution Increment
  - 2.3.4.1 #task# Design the Solution
  - 2.3.4.2 #task# Implement Developer Tests
  - 2.3.4.3 #task# Implement Solution
  - 2.3.4.4 #task# Run Developer Tests
  - 2.3.4.5 #task# Integrate and Create Build
- 2.5.4 #activity# Test Solution
  - 2.3.5.1 #task# Implement Tests
  - 2.3.5.2 #task# Run Tests
- 2.5.5 #activity# Finalize Product Documentation and Training
  - 2.4.7.1 #task# Develop Product Documentation
  - 2.4.7.2 #task# Develop User Documentation
  - 2.4.7.3 #task# Develop Support Documentation
  - 2.4.7.4 #task# Develop Training Materials
- 2.5.6 #activity# Prepare for Release
  - 2.5.6.1 #task# Review and Conform to Release Controls
  - 2.5.6.2 #task# Install and Validate Infrastructure
  - 2.5.6.3 #task# Develop Backout Plan
  - 2.5.6.4 #task# Develop Release Communications
- 2.5.7 #activity# Provide Product Training
  - 2.5.7.1 #task# Deliver end user Training
  - 2.5.7.2 #task# Deliver Support Training
- 2.5.8 #activity# Ongoing Tasks
  - 2.3.6.1 #task# Request Change
- 2.5.9 #activity# Deploy Release to Production
  - 2.5.9.1 #task# Package the Release
  - 2.5.9.2 #task# Execute Deployment Plan
  - 2.5.9.3 #task# Verify Successful Deployment
  - 2.5.9.4 #task# Execute Backout Plan (if necessary)
  - 2.5.9.5 #task# Deliver Release Communications

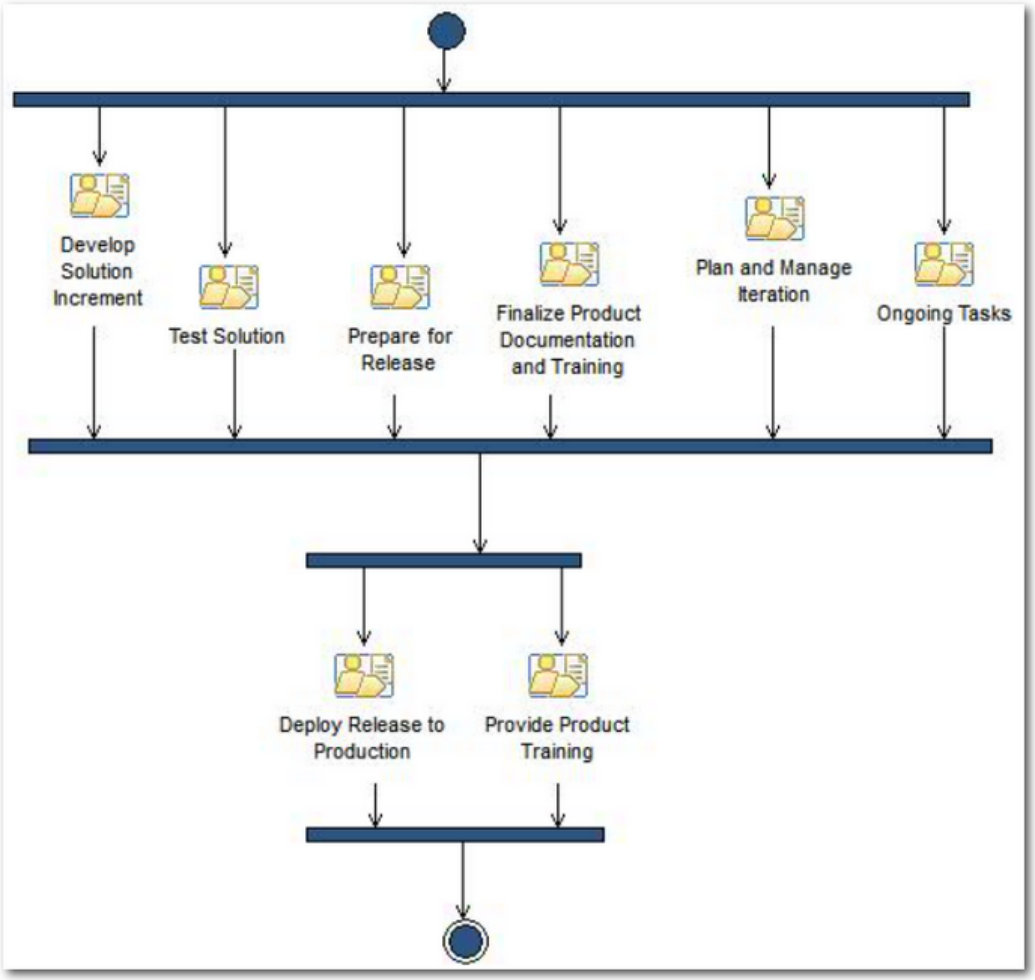
WBS

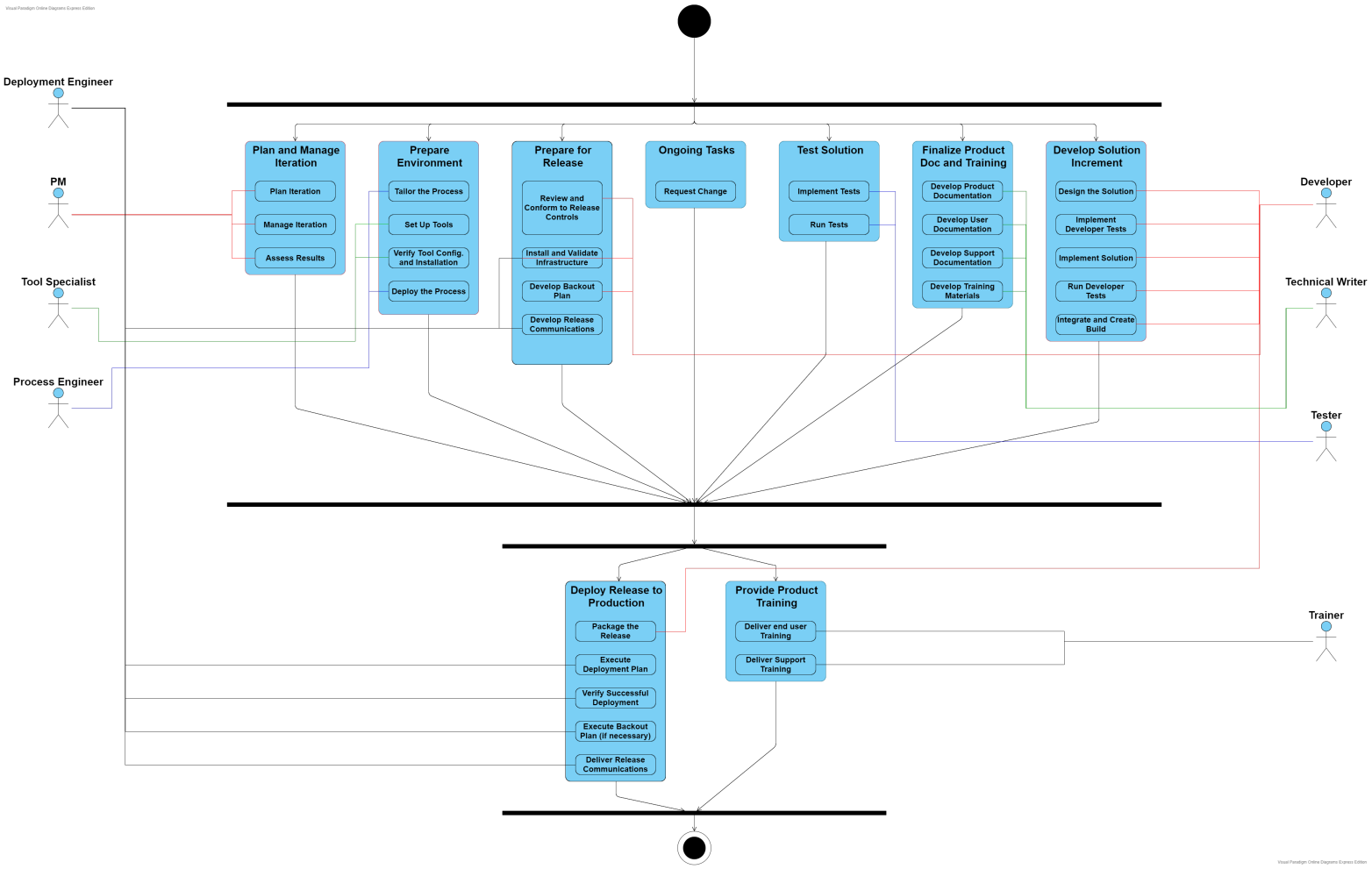
.

.

#iteration# Transition Iteration [1..n]

WBS





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

Description
In the Transition phase, the main objectives are to fine-tune the functionality, performance, and overall quality of the beta product from the end of the Construction phase.
The following table summarizes the Transition phase objectives and what activities address each objective:

Phase objectives	Activities that address objectives
Beta test to validate that user expectations are met	• Ongoing Tasks • Develop Solution Increment • Test Solution
Achieve stakeholder concurrence that deployment is complete	• Plan and Manage Iteration • Test Solution
Improve future project performance through lessons learned	• Plan and Manage Iteration

.

.
2.5.6 #activity# Prepare for Release

Summary
This activity prepares a product for release.

.

.
2.5.6.1 #task# Review and Conform to Release Controls

Summary
Release controls normally are specified by the deployment manager. These controls document the requirements to which all releases must conform before being placed into the production environment.

Purpose
The purpose of this task is to ensure that there are no (or minimal) negative impacts to existing production systems, products, services, and operations.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Developer	Mandatory: • #workproduct# Release Controls	
Additional: • #role#	Optional: • None	
Assisting:	External: • None	

.
Main Description
Release controls describe the minimum number of requirements that a software package must adhere to before being released into production. This is especially important if a development team is new or emerging, because they might not be aware of the great responsibilities a deployment manager has. In fact, a deployment manager is responsible to senior management for ensuring that nothing is placed into production that does not conform to the rigid controls designed to protect the IT organization's ability to successfully deliver IT services to internal and external customers.
Release controls typically consist of the following:

- Release or deployment plan
- Backout plan
- Release component definitions
- Release package integrity verification
- References to configuration items (CIs)
- Customer approval
- Ready for transfer to operations and support staff

Steps

- **Locate release controls**

  If the program's release controls are not readily available, the development team must engage the deployment manager and/or their deployment engineers to know where to find the release controls and be able to comply with them.

- **Review release controls**

  The development team should thoroughly review the release controls so that it understands what is expected before a release is accepted into the production environment. If the team has any questions or issues with the controls, team members should communicate directly with the deployment manager or the deployment engineer to understand the issues.

- **Ensure the team release conforms to the controls**

  Coordinated releases at the program level are very formal processes. They are formal for a very good reason - namely, the company's production environment could be corrupted and serious business ramifications could result, including:

  - Lost revenue
  - customer dissatisfaction
  - Fines resulting from legal noncompliance
  - Lost employee productivity

  All development team members that contribute to a release are expected to adhere to all the controls defined at the program level. Non-compliance could result in multiple impacts:

  - The entire release being abandoned, which could lead to customer or end user dissatisfaction
  - The results of multiple feature development Sprint/Iterations not being included in the release
  - Embarrassment on the part of the development team member that did not comply with the controls
  - Loss of funding for that development team

.

.

2.5.6.2 #task# Install and Validate Infrastructure

Summary
Any infrastructure components needed to support a release must be procured, installed, and tested.

Purpose
The purpose of this task is to ensure that all the necessary infrastructure components needed to support a successful release are in place.

Relationships.
.

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

• #role# Deployment Engineer • #role# Developer	• #workproduct# Deployment Plan	
Additional: • #role#	Optional: • #workproduct# Release Controls	
Assisting:	External: • None	

.

Main Description

A release package cannot be deployed to production if the environmental infrastructure within which the release will be run is not sufficiently built or tested. Whether the release is deployed as a "push" (where the application is deployed from a central point and proactively delivered to target locations) or a "pull" (where the application is made available at central point and pulled by a user at a time of their choosing), the infrastructure needed to support the application must be considered and implemented.

Some key aspects of installing and/or validating the desired infrastructure:

- Identify the requirements and components of the environment configuration
- Determine the lead times required to establish the infrastructure environments
- Procure and install the infrastructure components that are not yet available
- Test the newly installed infrastructure components
- Test the integration of newly installed components with the rest of the environmental configuration
- Validate other aspects of the infrastructure including:
  - Security components and their integration
  - Database connectivity and security
  - License management, as appropriate
  - Configuration management, in terms of configuration items (CIs)

Steps

- **Identify infrastructure needs**

  Identify and describe all the components of the infrastructure that are needed to support the upcoming release. These requirements should be based completely on the feature set that is about to be deployed, not on intended future needs.

- **Procure components**

  Determine how long it will reasonably take to procure the needed components and to engage the appropriate division in the organization to place the order. Be sure to work with the procurement agency to track the order and to identify any issues with that order. Ultimately, the development team, not the procurement agency, is responsible for ensuring that the correct infrastructure components are in place.

- **Schedule components for installation**

  After the procured components arrive, schedule to have them installed by the IT operations group(s) that control the production environment. The development team should be developing tests to confirm the correct installation at this time.

- **Install and test components**

  When the components have been installed, be prepared to run the validation tests developed in the previous step. These tests should not only verify the individual components' readiness, but also should validate their integration with each other and with legacy components.

- **Validate other component aspects**

  As the validation testing is underway, the development team also should consider how the newly installed components impact overall system security, whether or not database connectivity and security have been compromised, and what impact they have on the configuration management database that contains an inventory of CIs.

  Also reconcile any licensing issues for the new components with the division that is responsible for documenting and tracking enterprise licenses.

.

.

2.5.6.3 #task# Develop Backout Plan

Summary
This task results in a plan to be used by the production support organization to roll back the release if there is a problem during or after deployment.

Purpose
The purpose of this task is to develop the criteria, procedures, and responsibilities associated with rolling back a specific release into the production environment to prevent or minimize interruptions to other products or services.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Developer	Mandatory: • None	• #workproduct# Backout Plan
Additional: • #role# Deployment Engineer	Optional: • #workproduct# Deployment Plan	
Assisting:	External: • None	

.

Main Description

A rollback might be needed for a variety of reasons, including corruption of the production code base, inoperable components, an unplanned undesirable effect of the release on other production systems, an unhappy customer, etc. The Development team should provide the production support organization with a specific plan and decision criteria made available to them to avoid or minimize service interruptions.

Steps

- **Determine if backout plan exists**

  Determine whether the development team has a backout plan already written for a previous release. If so, part of that plan might be reusable. If this release is the development team's first, another development team with a similar feature set might have a plan that can be used as a starting point.

- **Develop the backout plan (if applicable)**

  If a backout plan does not exist, or one cannot be found to be used as a starting point, answer the questions documented in the Artifact: Backout Plan to start and develop a backout plan.

- **Update the backout plan (if applicable)**

  If a backout plan does exist that can be used as a baseline, review that plan and update, add, or delete information as necessary. When the plan is completed, it should reflect entirely the contents of the upcoming deployment only, not a release in the past or one in the future. In other words, the backout plan should be specific only to this release.

.

.

2.5.6.4 #task# Develop Release Communications

Summary

Stakeholders should be notified when a product (or feature set) is placed into production.

Purpose

The purpose of this task is to create the content and criteria to identify who is notified and to describe how they are notified after a release is deployed successfully to the production environment.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Deployment Engineer	Mandatory: • #workproduct# Deployment Plan	• #workproduct# Release Communications
Additional:	Optional:	

•	• #workproduct# Release Controls	
Assisting:	External: • None	

.

Main Description

When a release is pushed to production, all the stakeholders of that product should be notified that the event has happened and what the release means to each of the stakeholders. Often, the output of this task does not need to be created from scratch; for products that plan multiple releases, just updating the communique details for each release might be enough. However, if any of the stakeholder groups change, or there is a significant difference in the product distribution, more significant content might need to be developed.

In any case, communicating effectively to the end user community is important. A development team can develop high quality software, but if messaging to the stakeholders is conducted poorly or not at all, the end user experience might be degraded. By simply answering the questions "who, what, when, where, why, and how" in a format appropriate for each stakeholder group, a product release can become a more satisfying experience for all those involved.

Steps

- **Identify stakeholders for this release**

  The development team should know exactly which stakeholder groups will benefit from the upcoming release. First, identify the stakeholders for this release. Next, determine how each stakeholder group is expected to benefit from the release based on the components that will be delivered to production.

- **Draft communique for each stakeholder group**

  For each stakeholder group, document the following:

  - The features that will be deployed to production that those stakeholders are expected to benefit from
  - The business value that stakeholder group will obtain from the feature set being released
  - How, when, and where that stakeholder group will be able to access the new functionality and what special credentials or permissions are required
  - Any additional constraints or information that the stakeholder group should be aware of, such as availability restrictions, geographical restrictions, server limitations, regulatory requirements, etc.

- **Provide commiques to deployment manager**

  After drafting the communiques for each stakeholder group, the development team should provide those drafts to the deployment manager. Typically, release communications are consolidated and released at the program level. The deployment manager and deployment engineers normally are responsible for ensuring that all release communications are consistent and concise. The

deployment manager will determine the appropriate time to communicate information about the upcoming release to the appropriate stakeholders.

.

.

2.5.7 #activity# Provide Product Training

Summary
This activity provides product training.

.

.

2.5.7.1 #task# Deliver end user Training

Summary
In most cases, end users of a system require training to use the application effectively.

Purpose
The purpose of this task is to ensure that end users are properly trained to use the system to achieve the maximum business value.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Trainer	Mandatory: • #workproduct# Training Materials • #workproduct# User Documentation	
Additional: •	Optional: • #workproduct# Product Documentation	
Assisting:	External: • None	

.

Main Description
Often, a trainer will deliver training to end users; rarely will the development team deliver training because they are busy developing additional features for this or other systems. If a trainer is not available, the product owner might have to train the end users.

End user training usually consists of presentation slides, job aids, hands-on labs and exercises, or workshops that integrate these methods into an environment that the end users can understand and relate to.

Steps

- **Validate user training logistics**

  Before actually delivering the training to end users, ensure that you have all the necessary components of the training course, including:

  - Presentation slides
  - Handouts
  - Training aids
  - Hands-on labs and/or exercises
  - Quizzes or tests
  - Feedback and/or instructor assessment forms

  Ensure the classes have been scheduled correctly and attendees have been identified. Verify that the facilities where you will deliver the training have been properly booked and are furnished to accommodate the training content and audience. If possible, send a reminder notice to the attendees several days (or a week) before the delivery date.

- **Prepare for user training delivery**

  Review the training materials so that you are familiar with them. If you encounter any areas in which you are unsure or tentative about, engage experts from the development team or program who can help you understand the material. Then, update the training materials accordingly and create instructor notes to help you remember key points as you deliver the training. Instructor notes that you author will not only help you, but they might assist other instructors in the future who have not had the benefit of working with the development team or program.

- **Deliver user training and gather feedback**

  On the prescribed day(s) and time(s), deliver the training as planned. When each course is completed, solicit feedback from the attendees on the course material, its organization and flow, the learning objectives, how well the instructor delivered the content, and whether the course met their expectations.

- **Provide feedback to the program level**

  Consolidate the feedback from the courses delivered and provide it to the program manager or their delegate for review.

.

.

2.5.7.2 #task# Deliver Support Training

Summary
Personnel who support an application need training so they can perform their jobs effectively.

Purpose
The purpose of this task is to provide personnel at all levels of support (Tiers 1, 2, and 3) with the appropriate amount of training on the features that are being delivered as part of a specific Release.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Trainer	Mandatory: • #workproduct# Training Materials • #workproduct# Support Documentation	
Additional: •	Optional: • #workproduct# Product Documentation • #workproduct# User Documentation	
Assisting:	External: • None	

.

Main Description

Because a release ultimately will have to be supported by Help Desk or other technical personnel, a set of training materials designed to convey key concepts to those individuals must be developed and delivered by the development team or by a technically oriented trainer.

Delivery of this training might include presentation slides, job aids, lab exercises, or workshops designed to provide real-world scenarios to accelerate learning and retention. In addition, the training materials might be different for each level (tier) of support.

Steps

- **Validate support training logistics**

  Before actually delivering the training to support personnel, ensure that you have all the necessary components of the training course, including:

  - Presentation slides
  - Handouts
  - Training aids
  - Hands-on labs and/or exercises
  - Quizzes or tests
  - Feedback and/or instructor assessment forms

  Ensure that the classes have been scheduled correctly and that attendees have been identified. Also, validate that the facilities where you will deliver the training have been properly booked and are furnished to accommodate the training content and audience. If possible, send a reminder notice to the attendees several days (or a week) before the delivery date.

- **Prepare for support training delivery**

Review the training materials so that you are familiar with them. If you encounter any areas where you are unsure or tentative, engage experts from the development team or program who can help you understand the material. Then, update the training materials accordingly and create instructor notes to help you remember key points as you deliver the training. Instructor notes that you write will not only help you but might assist other instructors in the future who have not had the benefit of working with the development team or program.

- **Deliver support training and gather feedback**

  On the prescribed day(s) and time(s), deliver the training as planned. When each course is completed, solicit feedback from the attendees about the course material, its organization and flow, the learning objectives, how well the instructor delivered the content, and whether the course met their expectations.

- **Provide feedback to the program**

  Consolidate the feedback from the courses delivered and provide it to the program manager or their delegate for review.

.

.

2.5.9 #activity# Deploy Release to Production

Summary
This activity results in the release of a set of integrated components into the production environment.
.

.

2.5.9.1 #task# Package the Release

Summary
Each release should be built and packaged in a standard, controlled, and repeatable manner.

Purpose
The purpose of this task is to render a complete, deployable package capable of being released into the production environment by the deployment engineer.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Developer	Mandatory: • #workproduct# Deployment Plan • #workproduct# Release Controls	• #workproduct# Release
Additional: • #role# Deployment	Optional: • None	

Engineer		
Assisting:	External: • None	

.

Main Description

The key activities normally used to package a release:

- Assemble the components and integrate them through a normal (i.e., continuous integration) or release build script
- Install the release package in one or more test environments and verify its integrity
- Tag the elements of the release package in the code base to create a baseline
- Package appropriate documentation to accompany the release:
  - Deployment plan
  - Build plan, procedures, and scripts
  - Backout plan
  - Relevant licensing information
  - Relevant infrastructure information
  - Release communiques

Steps

- **Assemble components**

  Question all the developers on the development team to determine which components are ready for packaging. Only package those components that were completed and accepted during the previous feature development sprint/iterations. Components that were not finished or not accepted should not be bundled, unless the customer has granted an exception or they are infrastructure-related components.

- **Test the release**

  After the components have been packaged and built, that executable should be installed and run in a test environment that mimics the production environment. A "staging" environment usually is maintained for this purpose. Testing typically includes a "smoke test" in which key features are exercised to highlight any unplanned behavior.

- **Tag source code repository**

  In the team's configuration management (CM) tool, tag all the components that went into the release package so that the package can be reconstructed at a later date, if needed. This tag is known as the release "baseline."

- **Package release documentation**

  Gather all the product, user, and support documentation developed earlier in the production release sprint/iteration and add it to the release package.

- **Deliver release package**

  When the entire release package, including documentation, is ready, deliver it to the deployment manager and the release team in a timely manner. Be prepared to answer questions from the deployment engineer, especially questions about conformity to release controls.

.

.

2.5.9.2 #task# Execute Deployment Plan

Summary
The deployment plan has all the unique instructions necessary to roll out a release.

Purpose
The purpose of this task is to ensure that the rollout is based on clear, validated, repeatable instructions and to reduce the risk of a deployment issue.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Deployment Engineer	Mandatory: • #workproduct# Deployment Plan • #workproduct# Infrastructure • #workproduct# Release	
Additional: • #role# Developer	Optional: • #workproduct# Release Controls	
Assisting:	External: • None	

.
Main Description
This task is straightforward: follow the procedures in the Deployment Plan for the rollout of a specific product release. If the deployment plan does not exist or it is poorly constructed, this task might be much more difficult.

The main point here is that to achieve a high probability of success, the development team should have previously developed a detailed plan that organizes and articulates all the unique instructions for deploying that particular release. Because an experienced deployment engineer normally executes this task, they might be able to overcome any missing deployment procedures or content. However, that is not an excuse for a development team to not develop the plan's contents.

Steps

- **Review deployment plan**

Review the contents of the deployment plan to ensure that the production environment has all the dependent components installed and that the system is in the correct state. Also ensure that the release window is ready to be achieved.

- **Release code**

  When the preliminary review has been completed and the release window has started, deploy the release package into production. Depending on the release script and the size of the package, this installation might take anywhere from a few minutes to a few hours.

  Monitor the release as the release script is run. Be prepared to terminate the script and back out the release if significant errors are encountered.

.

.

2.5.9.3 #task# Verify Successful Deployment

Summary
Determine whether the rollout of a particular release into production has been successful or not.

Purpose
The purpose of this task is to confirm that a release has not caused any unintentional interruptions to service in the production environment.

Relationships.
.

Roles	Inputs	Outputs
Primary: • #role# Deployment Engineer	Mandatory: • #workproduct# Release	
Additional: • #role# Developer • #role# Product Owner	Optional: • #workproduct# Deployment Plan	
Assisting:	External: • None	

.

Main Description
Using the success criteria documented either in the deployment plan or in the backout plan, the deployment engineer, in collaboration with the development team, will determine whether the rollout can be declared a success or not.

If the deployment is successful, the previously prepared release communiques should be delivered. If the deployment is unsuccessful, then the backout plan should be invoked.

Steps

- **Test production release**

  In this step, automated smoke tests should be run to determine whether key components were deployed successfully. These tests should be brief but revealing enough to quickly determine the validity of the deployment.

- **Run manual tests**

  If the automated smoke tests are successful, run several complex manual tests to simulate key end user behavior. These tests should be executed by development team members or stakeholders recruited specifically for this purpose.

- **Determine if release should be reversed**

  In some situations, problems with the release might be encountered but are not serious enough to reverse the deployment. If the problem(s) associated with the release can be fixed easily, and if they are not detrimental to the production environment, an emergency bug fix (EBF) might be the answer. In that case, the release is not backed out; rather, an EBF is scheduled to be executed as soon as possible.

.

.

2.5.9.4 #task# Execute Backout Plan (if necessary)

Summary
If a particular release into production goes wrong, the plan for cleanly reversing that deployment is executed.

Purpose
The purpose of this task is to remove a specific release quickly and seamlessly from the production environment because the release has caused problems or because it has been determined by the stakeholder community as unfit for service.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Deployment Engineer	Mandatory: • #workproduct# Backout Plan • #workproduct# Release	
Additional: • #role# Developer	Optional: • None	
Assisting:	External: • None	

.

Main Description
Assuming a backout plan is available for this release, the deployment engineer (or development team) will follow the instructions for reversing the installation of the product into production, if there is a problem. While the plan might have been written with good intentions, sometimes key procedures are missing or have not been thought out. The team backing out the release should be aware that blindly following the backout plan might not be the best approach. It is best to consider the unique circumstances within which the deployment has failed and rely on common sense and experience when executing the backout plan.

Steps

- **Identify release problem(s)**

  In the event that the release to production experiences problems, either during or after deployment, the backout plan should be invoked. However, the deployment engineer (or development team) must understand where the release went wrong so that the code can be fixed before the next release attempt. This is a critical step, but it should be done quickly so that the problematic release can be reversed before significant damage is done to the production environment.

  Log the issues as critical defects as soon as possible and assign those defects to the appropriate team members for resolution.

- **Backout the release**

  Following the instructions in the backout plan, reverse the deployment. However, be aware that the backout plan instructions are a guide and should not always be taken literally. This interpretation is due to the fact that not every problematic condition can be documented in advance and because each real-world situation might be slightly different.

- **Determine if the backout was successful**

  Ascertain whether the reversal was successful. If not, key members of the release team, development team, or program level agile system team might need to be engaged to find and fix the problem(s).

- **Communicate the backout**

  Ensure that all interested parties are aware of the failed release. Because releases often take place at odd hours to minimize end user impact, use beeper-based notifications judiciously. In most cases, an email to key stakeholders (such as the product owner and program manager) might suffice. Following up with a phone call also might be warranted.

.

.
2.5.9.5 #task# Deliver Release Communications

Summary
After a product release has been declared a success and is available for use, all relevant stakeholders should receive a communique stating that fact.

Purpose

The purpose of this task is to ensure that the widest possible distribution of accurate information regarding a particular release takes place.

Relationships.

.

Roles	Inputs	Outputs
Primary: • #role# Deployment Engineer	Mandatory: • #workproduct# Release Communications	
Additional: • #role# Developer	Optional: • #workproduct# Release	
Assisting:	External: • None	

.

Main Description

This task represents the distribution of communiques that were prepared beforehand as part of the release communications artifact. Although the development team is responsible for preparing the communications, the responsibility for sending the communiques normally is assigned to the deployment engineer, if that is the organizational protocol.

Steps

- **Validate the communiques**

  Ensure that the communiques designated for this release are correct, complete, and reflect all the stakeholder groups that must be informed.

- **Send release communications**

  Using a pre-planned script, send the communiques in the manner designated in the Artifact: Release Communications. Most of the time, the preferred communication mechanism will be by email, but other methods could include beeper notification, telephone calls, a posting to an internal release website, live or pre-recorded presentations by senior management, etc.

- **Validate communications were received**

  If the communiques were sent by email, it may be prudent to request a receipt of delivery to ensure that the notices were received by the intended audience.

.

.