PACE UNIVERSITY
Seidenberg School of Computer Science and Information Systems

CS631G Software Verification

CONFIGURATION, CHANGE AND RELEASE MANAGEMENT (W9)

Class instructor: Yuri Chernak, PhD

Outline

Section	Outline
Part I. Introduction to ITIL Framework	• ITIL process and lifecycle phases • ITIL Service Transition explained
Part II. Software Configuration Management	• Configuration Management overview • Configuration Management activities • CMMI definition of Configuration Management • IEEE Std 828-1998 – SCM Plans • Version Management explained • Code Check-in and Check-out Illustrated • Code branching and merging
Part III. Change Management	• Change Management overview • Change and Release Management flow • Change Manager role • Change Impact Analysis explained
Part IV. Release and Deployment Management	• Release and Deployment Management overview • ITIL types of releases explained • Production Acceptance Gates framework • Release deployment planning

Class Objectives

The objective of this class is to introduce the IT Information Library (ITIL) and discuss in detail the software configuration, change, and release management practices.

At the end of this class you will:

- understand the ITIL model and its Service Transition phase;
- learn Software Configuration Management practices;
- learn Change Management practices;
- learn Release Management practices;
- understand the relationships between the above processes.

Part I

Part I.
Introduction to ITIL

What is ITIL?

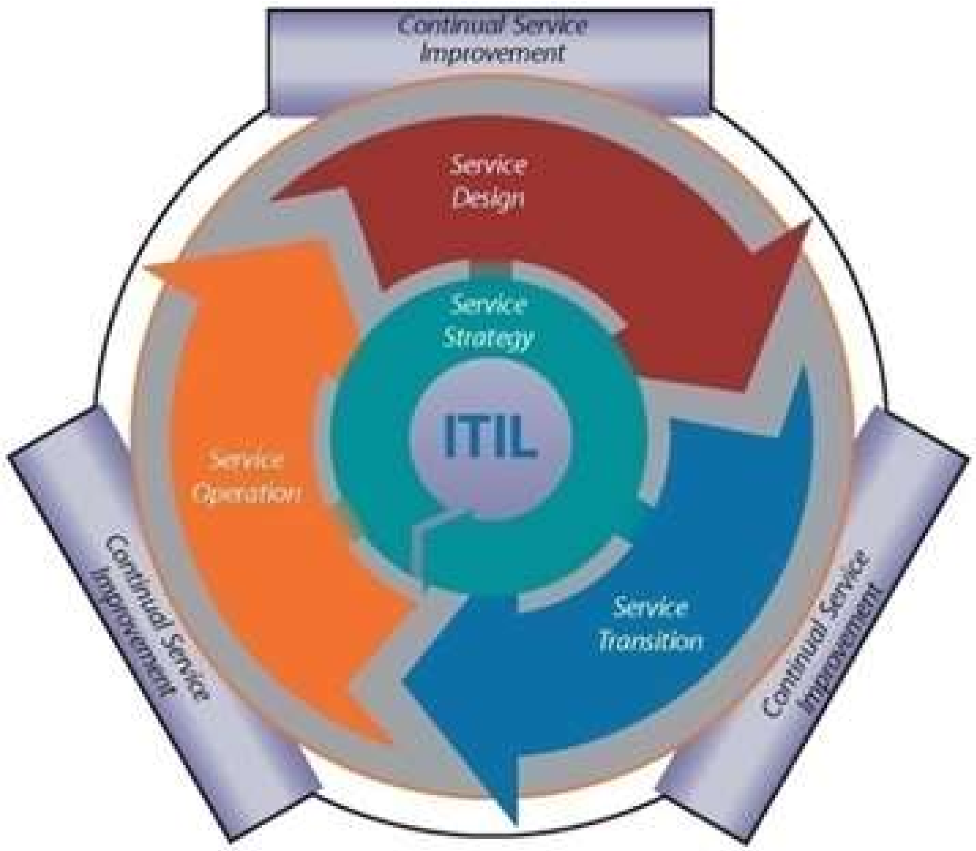
- IT organizations are continuously challenged to deliver better IT services at lower cost in a turbulent environment.
- Several management frameworks have been developed in the IT Industry to cope with this challenge.
- The IT Infrastructure Library (ITIL) is the most common framework that focuses on aligning IT services with the needs of business.
- The ITIL framework was originally developed in the 1980s in the UK at the same time when the CMM framework was being developed in the US.
- ITIL is maintained by the itSMF (the IT Service Management Forum) who publishes new ITIL editions (current ITIL 4 Edition, 2019).

ITIL Lifecycle Phases

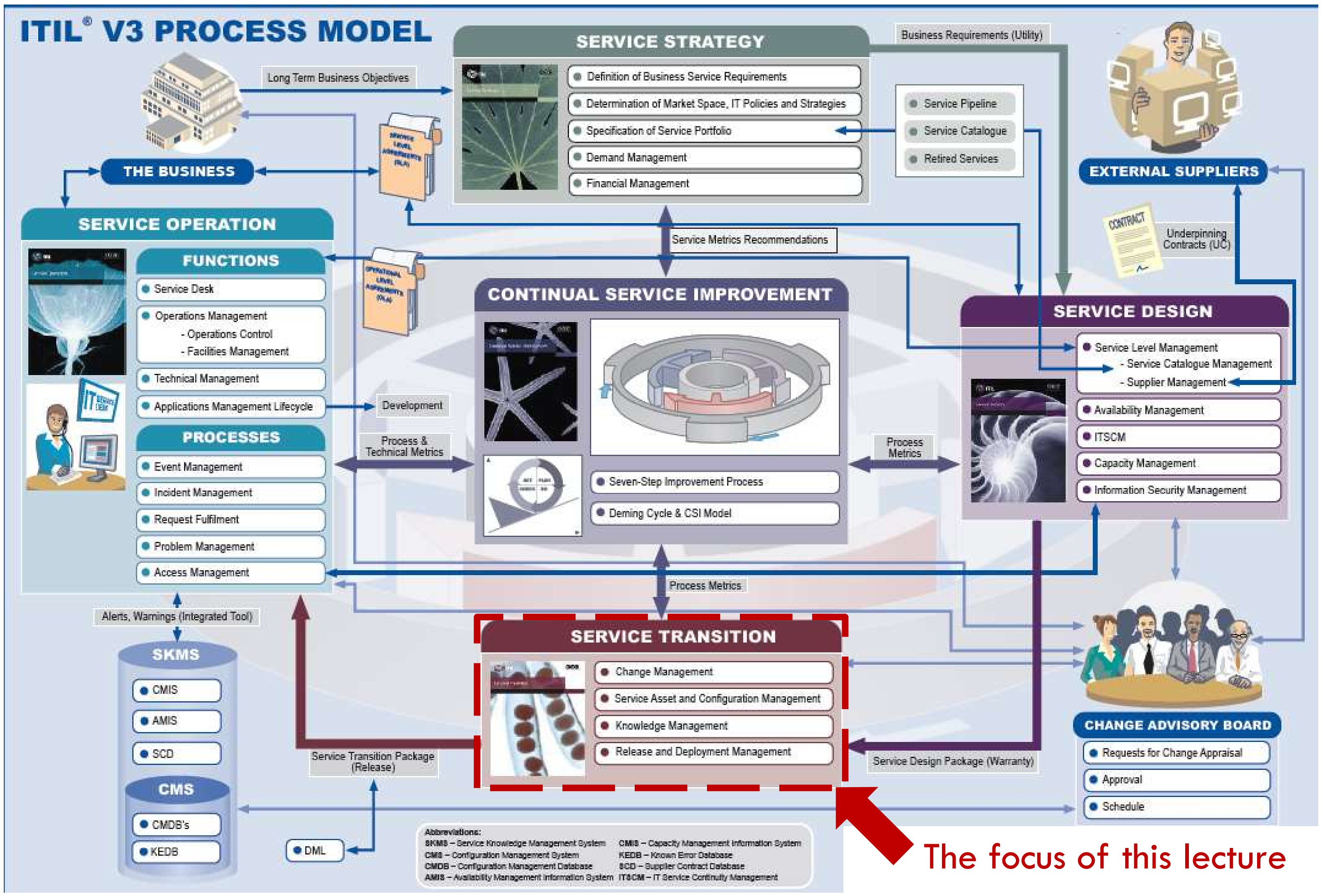
ITIL defines the Service Lifecycle as five phases:

- Service Strategy
- Service Design
- Service Transition
- Service Operation
- Continual Service Improvement

The focus of this lecture is Service Transition, which Includes Configuration Management, Change Management, and Release Management.


Continual Service Improvement
Service Design
Service Strategy
ITIL
Service Operation
Service Transition
Continual Service Improvement
Continual Service Improvement

The ITIL Process Model


ITIL® V3 PROCESS MODEL
Long Term Business Objectives
THE BUSINESS
SERVICE STRATEGY
Definition of Business Service Requirements
Determination of Market Space, IT Policies and Strategies
Specification of Service Portfolio
Demand Management
Financial Management
Business Requirements (Utility)
Service Pipeline
Service Catalogue
Retired Services
EXTERNAL SUPPLIERS
SERVICE LEVEL AGREEMENTS (SLA)
OPERATIONAL LEVEL AGREEMENTS (OLA)
CONTRACT
Underpinning Contracts (UC)
Service Metrics Recommendations
SERVICE OPERATION
FUNCTIONS
Service Desk
Operations Management
- Operations Control
- Facilities Management
Technical Management
Applications Management Lifecycle
PROCESSES
Event Management
Incident Management
Request Fulfilment
Problem Management
Access Management
IT Service Desk
Development
Process & Technical Metrics
CONTINUAL SERVICE IMPROVEMENT
Seven-Step Improvement Process
Deming Cycle & CSI Model
Process Metrics
SERVICE DESIGN
Service Level Management
- Service Catalogue Management
- Supplier Management
Availability Management
ITSCM
Capacity Management
Information Security Management
Process Metrics
Alerts, Warnings (Integrated Tool)
SKMS
CMIS
AMIS
SCD
CMS
CMDB's
KEDB
DML
Service Transition Package (Release)
SERVICE TRANSITION
Change Management
Service Asset and Configuration Management
Knowledge Management
Release and Deployment Management
Service Design Package (Warranty)
CHANGE ADVISORY BOARD
Requests for Change Appraisal
Approval
Schedule
Abbreviations:
SKMS – Service Knowledge Management System
CMS – Configuration Management System
CMDB – Configuration Management Database
AMIS – Availability Management Information System
CMIS – Capacity Management Information System
KEDB – Known Error Database
SCD – Supplier Contract Database
ITSCM – IT Service Continuity Management

The focus of this lecture

ITIL — Service Transition Overview

Service Transition Purposes

- Plan and manage the capacity and resources required to package, build, test and deploy a release into production and establish the service specified in the customer and stakeholder requirements.
- Provide a consistent and rigorous framework for evaluating the service capability and risk profile before a new or changed service is released or deployed.
- Establish and maintain the integrity of all identified service assets and configurations as they evolve through the Service Transition stage.
- Provide good-quality knowledge and information so that Change, Release, and Deployment Management can expedite effective decisions about promoting a release through the test environments and into production.
- Provide efficient repeatable build and installation mechanisms that can be used to deploy releases to the test and production environments and be rebuilt, if required, to restore service.
- Ensure that the service can be managed, operated, and supported in accordance with the requirements and constraints specified within the Service Design.

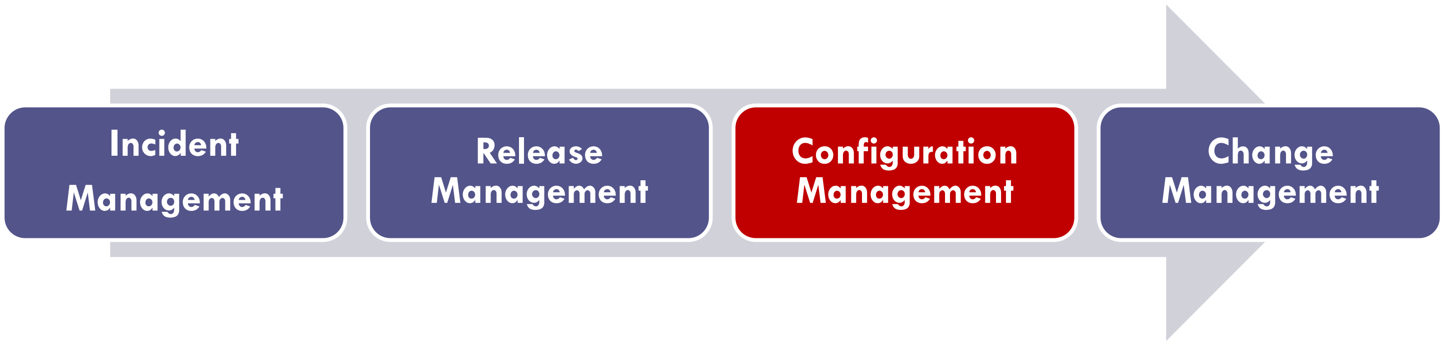
ITIL — Service Transition Processes

Covered in this lecture

Process Name	Process Objective
Transition Planning and Support	To plan and coordinate the resources to successfully establish a new or changed service into production within the predicted cost, quality and time estimates.
Change Management	To control the lifecycle of all Changes. The primary objective of Change Management is to enable beneficial Changes to be made, with minimum disruption to IT services.
Service Asset and Configuration Management	To maintain information about Configuration Items required to deliver an IT service, including their relationships.
Release and Deployment Management	To plan, schedule, and control the movement of releases into test and live environments. The primary goal of Release Management is to ensure that the integrity of the live environment is protected and that the correct components are released.
Service Validation and Testing	To ensure that deployed Releases and the resulting services meet customer expectations, and to verify that IT operations is able to support the new service.
Knowledge Management	To gather, analyze, store and share knowledge and information within an organization. The primary purpose of Knowledge Management is to improve efficiency by reducing the need to rediscover knowledge.

Part II

Part II.
Software Configuration Management



What is Configuration Management?

- Software systems are constantly changing during development and use.
- Configuration Management (CM) is concerned with the policies, processes, and tools for managing changing software systems.
- You need CM because it is easy to lose track of what changes and component versions have been incorporated into each system version.
- CM is essential to enable project teams to control changes made by different developers.

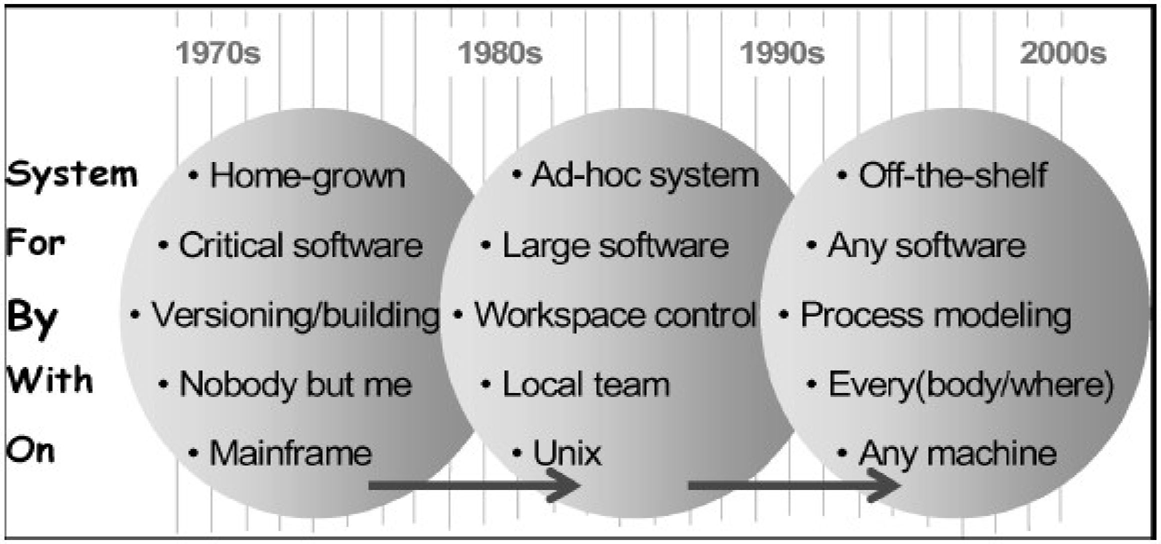
Configuration Management is the process of controlling and monitoring changes to work products. This includes many activities discussed on the next slides.

The IT industry common reference - IEEE Std. 828: “Software Configuration Management Plans”.

Evolution of Configuration Management

- SCM as a discipline emerged in the late 70s and early 80s, soon after the so called "software crises," when it was realized that software development consisted of more than just programming. Issues like architecture, building, evolution and so on were also important, time-consuming parts of the process.
- The focus of SCM has changed over the years as it has tried to address the different issues of software development.

IEEE Std.828: Application Level SCM



ITIL SCM

1990s-2000s

Enterprise Level:
Service Asset & Configuration Management

Configuration Management Terminology

Term	Definition
Baseline	A baseline is a collection of component versions that make up a system. Baselines are controlled, which means that the versions of the components making up the system cannot be changed. This means that it is always possible to recreate a baseline from its constituent components.
Codeline (a.k.a. codebase)	A codeline is a set of versions of a software component and other configuration items on which that component depends.
Branching	The creation of a new codeline from a version in an existing codeline. The new codeline and the existing codeline may then develop independently.
Configuration (version) control	The process of ensuring that versions of systems and components are recorded and maintained so that changes are managed and all versions of components are identified and stored for the lifetime of the system.
Configuration item or Software Configuration item	A work product that is treated as a single entity for the purpose of configuration management. Configuration items have a unique name. Anything associated with a software project (design, code, test data, document, etc.) that has been placed under configuration control. There are often different versions of a configuration item.

Configuration Management Terminology

Term	Definition
Change Request	A formal report issued by a user or a developer to request a modification of a configuration item.
Merging	The creation of a new version of a software component by merging separate versions in different codelines. These codelines may have been created by a previous branch of one of the codelines involved.
Version	Identifies the state of a configuration item at a given point in time. An instance of a configuration item that differs, in some way, from other instances of that item. Versions always have a unique identifier.
Promotion	A version that has been made available to other developers.
Release	A version of a system that has been released to customers (or other users in an organization) for use.
Repository	A shared database of versions of software components and meta-information about changes to these components.
System building	The creation of an executable system version by compiling and linking the appropriate versions of the components and libraries making up the system.
Workspace	A private work area where software can be modified without affecting other developers who may be using or modifying that software.

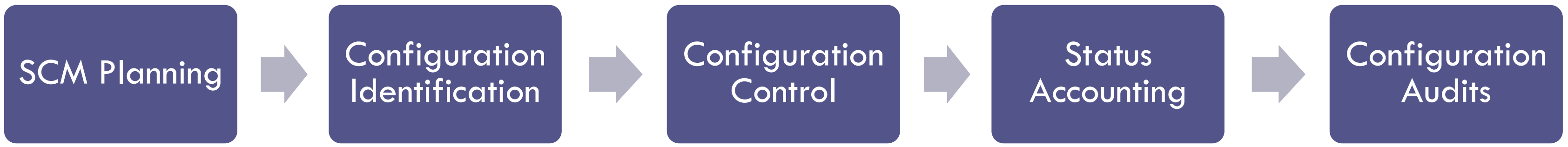
Configuration Management Definition

IEEE Std. 610 Definition – "Configuration Management" term

Software Configuration Management (SCM) is a discipline applying technical and administrative direction and surveillance to:

- **identify and document** the functional and physical characteristics of a configuration item,
- **control** changes to those characteristics,
- **record and report** change processing and implementation status, and verify compliance with specified requirements.

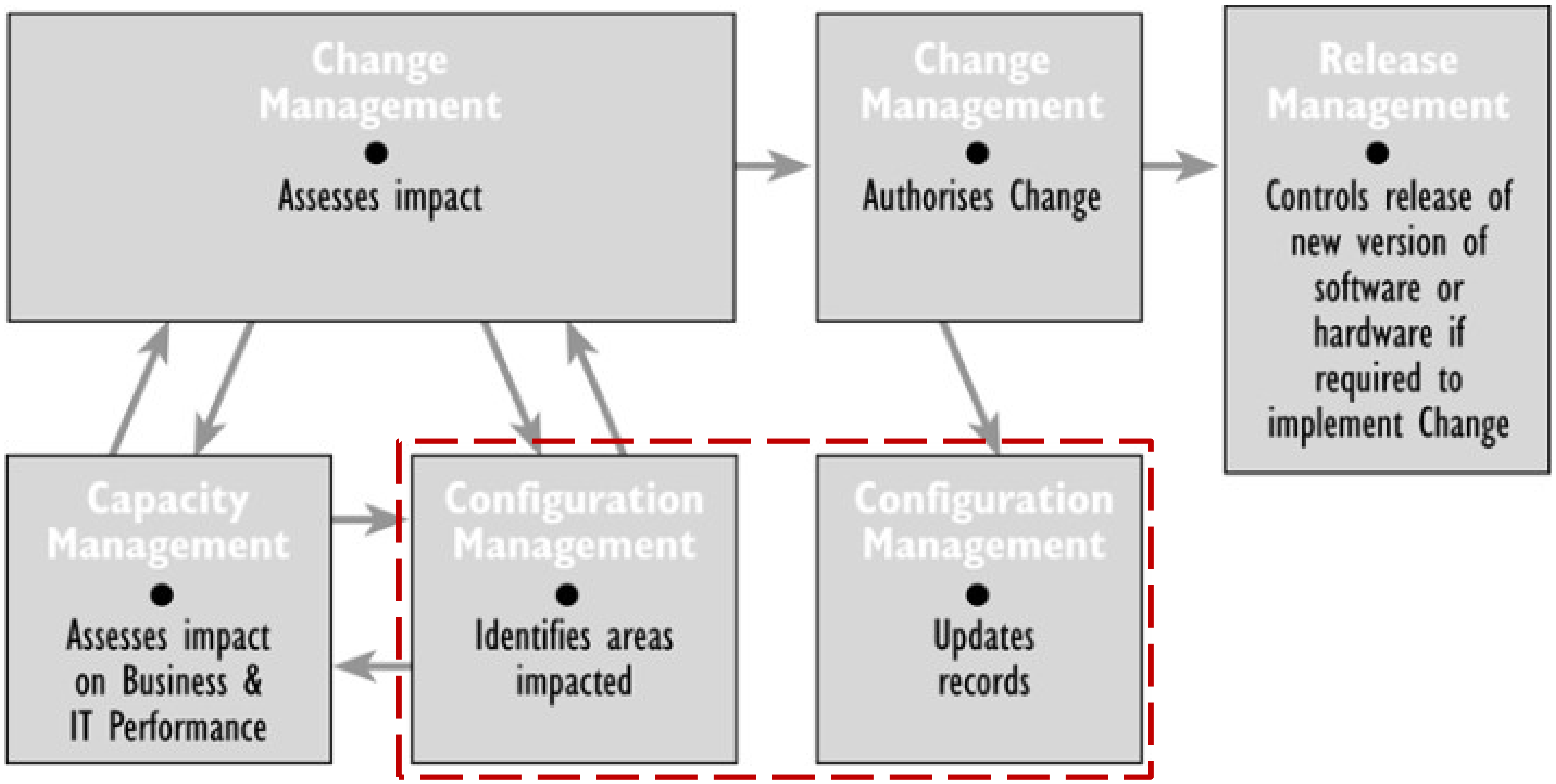
Conventional SCM Activities



Configuration Management Activities

Activities	Description
SCM Planning	• SCM Planning is a set of activities to establish an SCM policy for an organization, SCM plan for a given development team, as well establish a Configuration Manager role. • The Configuration Manager is responsible for the SCM process being defined, effective and followed.
Configuration Identification	• An element of configuration management, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation.
Configuration Control	• An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.
Status Accounting	• An element of configuration management, consisting of the recording and reporting of information needed to manage a configuration effectively. • This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes.
Configuration Audits and Review	• Configuration audits are broken into functional and physical configuration audits. They occur either at delivery or at the moment of effecting the change. • A functional configuration audit ensures that functional and performance attributes of a configuration item are achieved, while a physical configuration audit ensures that a configuration item is installed in accordance with the requirements of its detailed design documentation.

SCM Relationship with Other ITIL Processes



Examples of Work Products Under Configuration Management

Any of the work products below can be qualified as "configuration items":

Documentation	Software/Hardware	Tools
• Product specifications • Product technical publications • Plans (e.g. deployment) • User stories • Iteration backlogs • Process descriptions • Requirements • Architecture documentation and design data, etc.	• Code and libraries • Hardware and equipment • Installation logs • Product data files • Test scripts	• Compilers • Test tools • Tool configurations

SCM is a Process Area Defined in CMMI

Software Configuration Management is one of the process areas defined in the CMMI framework (*Capability Maturity Model*).

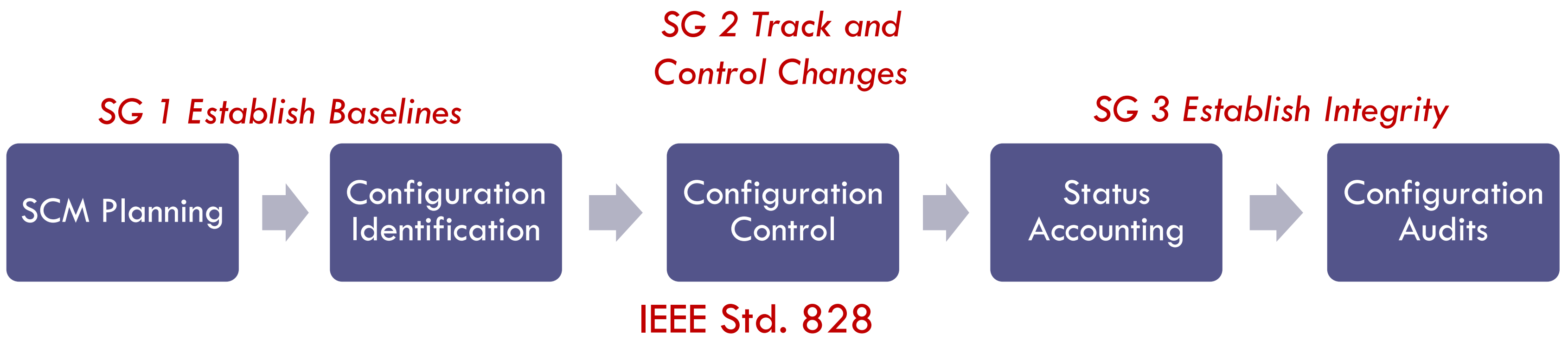
SCM Purpose (*as defined in CMMI*)
The purpose of Configuration Management (CM) is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.

The Configuration Management process area involves the following activities:

- Identifying the configuration of selected work products that compose baselines at given points in time.
- Controlling changes to configuration items.
- Building or providing specifications to build work products from the configuration management system.
- Maintaining the integrity of baselines.
- Providing an accurate status and current configuration data to developers, end users, and customers.

CM Practices Defined in CMMI

Process Goals	Specific Practices
SG 1 Establish Baselines	SP 1.1 Identify Configuration Items
	SP 1.2 Establish a Configuration Management System
	SP 1.3 Create or Release Baselines
SG 2 Track and Control Changes	SP 2.1 Track Change Requests
	SP 2.2 Control Configuration Items
SG 3 Establish Integrity	SP 3.1 Establish Configuration Management Records
	SP 3.2 Perform Configuration Audits



Configuration Management Roles

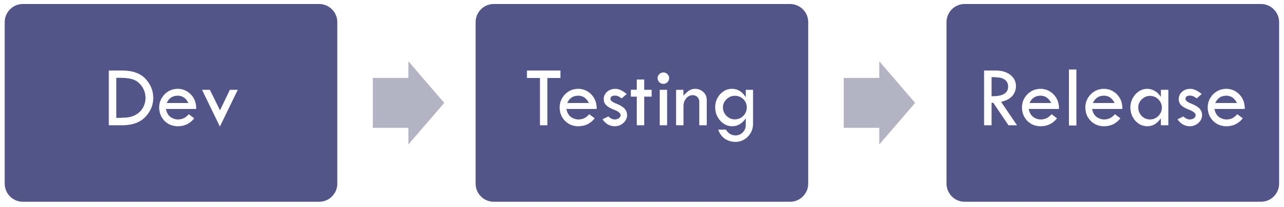
- Configuration Management is an important project function that involves many different tasks and participants in the project.
- Performing CM activities includes the following roles:

Role	Role Description
Configuration Manager	This role is responsible for identifying configuration items, and may also be responsible for defining the procedures for creating baselines and promotions. This role is commonly combined with the role of a technical lead on the project.
Change Control Board (a.k.a. Change Advisory Board in ITIL)	CCB (CAB in ITIL) is responsible for approving or rejecting change requests based on project goals. This role can also be involved in assessing the change impact and cost.
Developer	This role creates promotions triggered by change requests. The main activity is to check-in changes to the codebase repository.
Librarian	This role is responsible for establishing the configuration management library and for maintaining the library's integrity. This role can be combined with the Configuration Manager role.
Auditor	This role is responsible for evaluating promotions for releases to ensure code consistency and completeness.

CM Activities vs. Development Phases

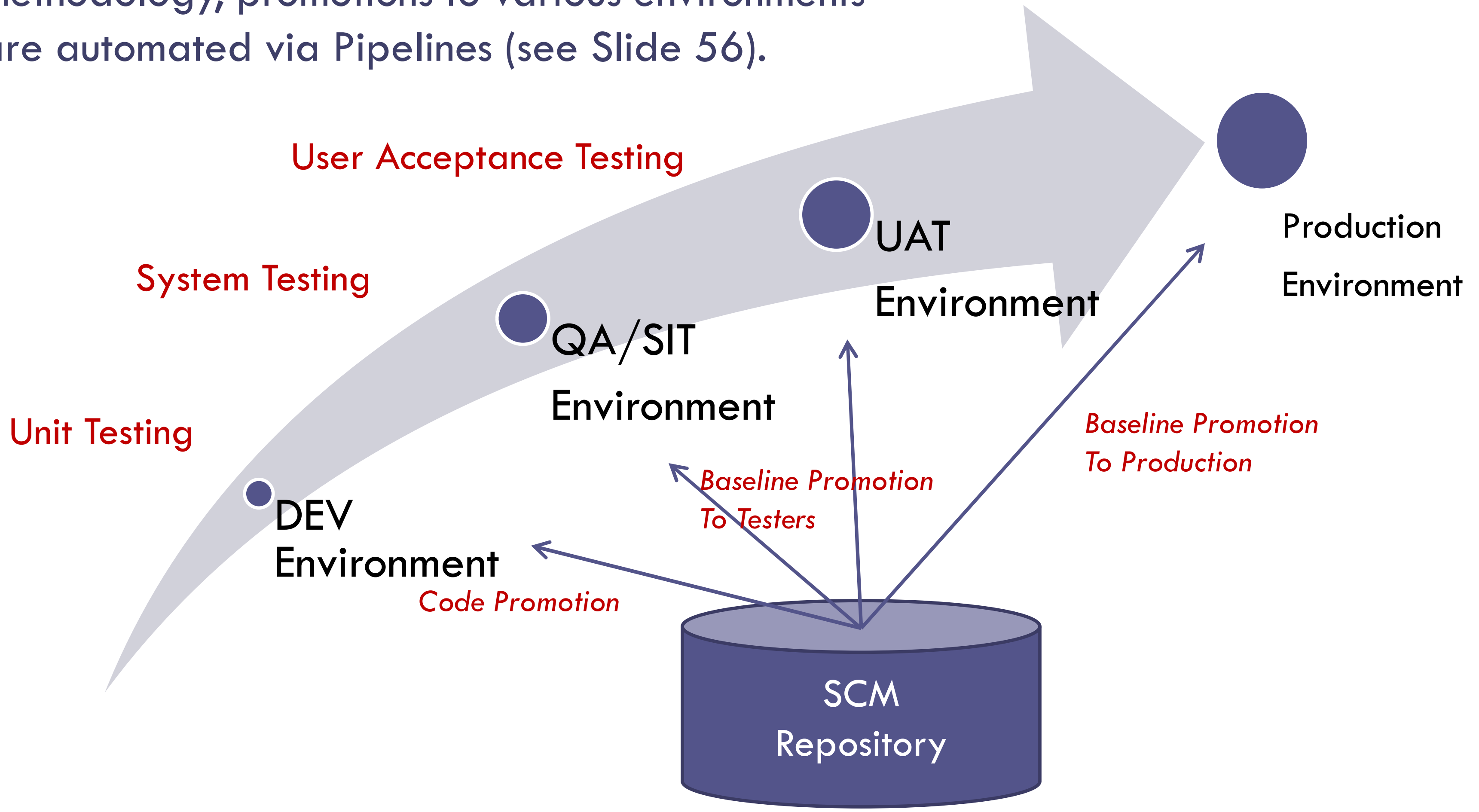
- A development phase where the development team is responsible for managing the software configuration and new functionality is being added to the software.
- A system testing phase where a version of the system is released internally for testing.
  - *No new system functionality is added. Changes made are bug fixes, performance improvements and security vulnerability repairs.*
- A release phase where the software is released to customers for use.
  - *New versions of the released system are developed to repair bugs and vulnerabilities and to include new features.*

Development Phases	Dev	Testing	Release
CM Activities	• Establish CM repository • Create new revisions and baselines	Promote new baselines to testing environments.	Deploy approved baselines into a production environment.



CM Promotions vs. Test Levels

When a project team follows the DevOps methodology, promotions to various environments are automated via Pipelines (see Slide 56).



Planning Configuration Management

- A Configuration Manager, along with a Project Manager, should plan configuration management activities.
- The key points in planning configuration management include:
  - defining the configuration management process, policies for change control and version management;
  - defining and assigning CM process roles, i.e., who takes responsibility for the CM procedures and creation of baselines;
  - defining change review and acceptance criteria;
  - defining release criteria;
  - defining the types of documents to be managed and a document naming scheme for the configuration items;
  - identifying the tools that should be used to support the CM process and any limitations on their use;
  - defining the structure of the configuration management repository used to manage software versions and configurations.

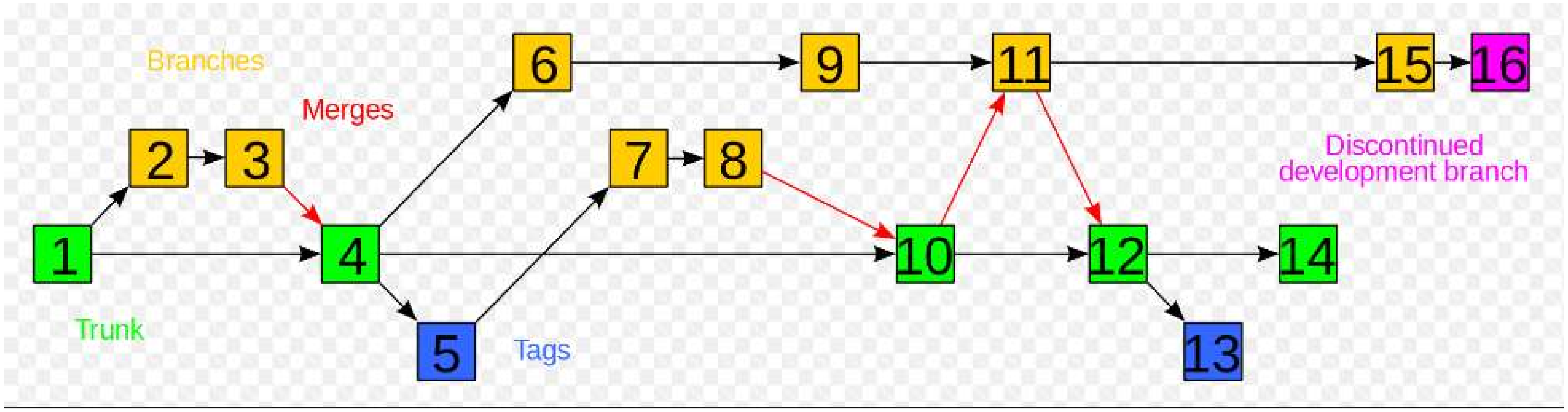
IEEE Std 828-1998 – SCM Plans

SCM planning information should include the following topics:

SCM Plan Topics	Description
Introduction	Describes the Plan's purpose, scope of application, key terms, and references.
SCM management	(Who?) Identifies the responsibilities and authorities for accomplishing the planned activities.
SCM activities	(What?) Identifies all activities to be performed in applying to the project.
SCM schedules	(When?) Identifies the required coordination of SCM activities with the other activities in the project.
SCM resources	(How?) Identifies tools and physical and human resources required for execution of the Plan.
SCM plan maintenance	Identifies how the SCM Plan will be kept current while in effect.

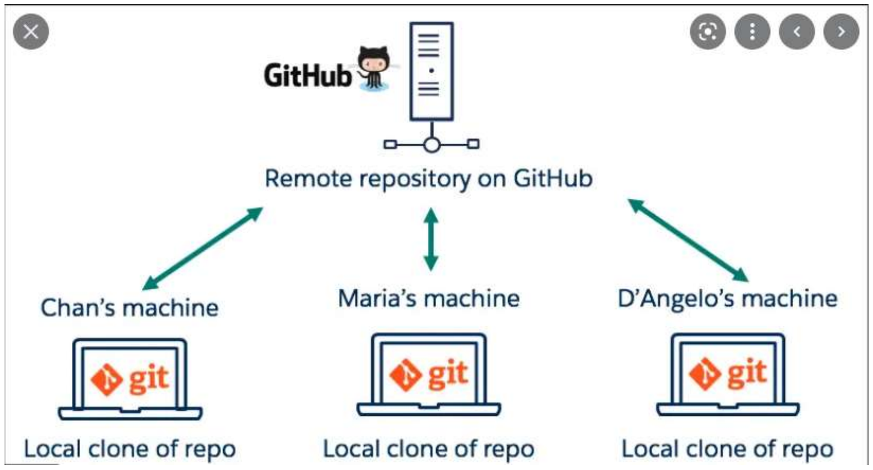
Version Management

- Version management (VM) is the process of keeping track of different versions of software components or configuration items and the systems in which these components are used.
- It also involves ensuring that changes made by different developers to these versions do not interfere with each other.
- A codeline is a sequence of versions of source code with later versions in the sequence derived from earlier versions.
- Codelines normally apply to components of systems, so that there are different versions of each component.

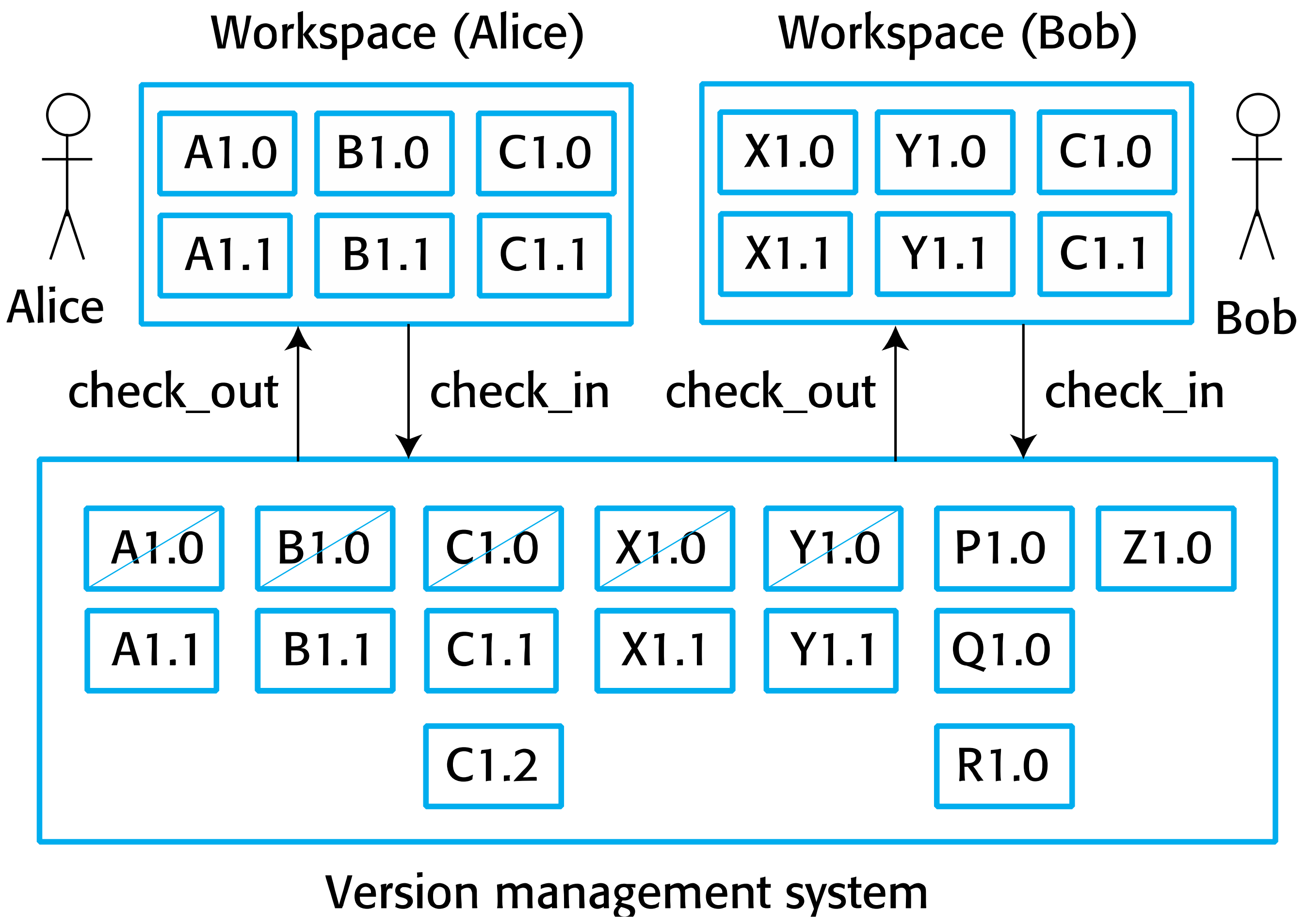


Central Code Repository

- Developers check-out components or directories of components from the project repository into their private workspace and work on these copies in their private workspace.
- When their changes are complete, they check-in the components back to the repository.
- If several people are working on a component at the same time, each developer checks it out from the repository. If a component has been checked out, the version control system warns other users wanting to check out that component that it has been checked out by someone else.

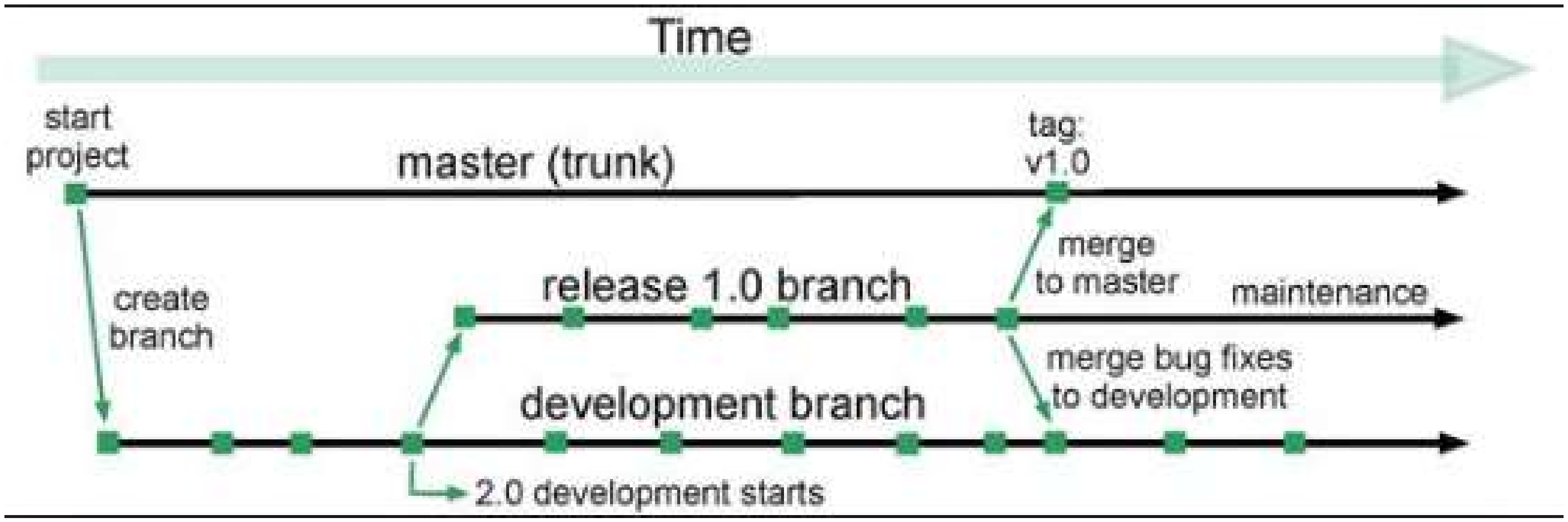


Code Check-in and Check-out Illustrated



Trunk and Branching Example

- The trunk (a.k.a. mainline) is usually meant to be the base of a project on which development progresses.
- Commonly, developers split a branch off the trunk, implement changes in that branch and merge the changes back into the trunk when the branch has proven to be stable and working.



Approaches to Creating Branches

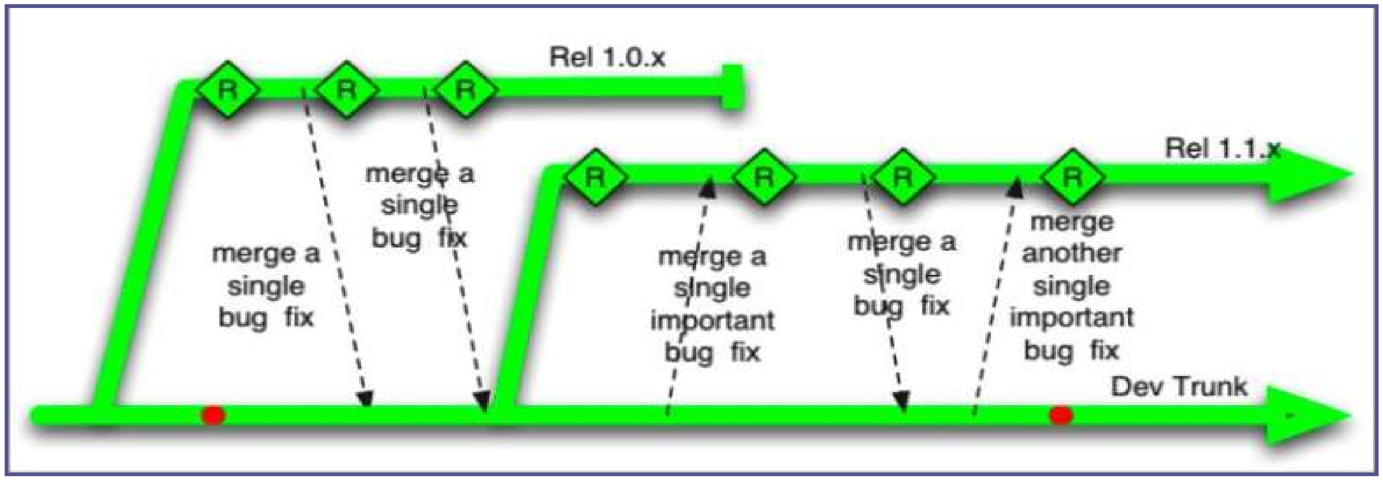
There are three common approaches to creating braches in a source-code management system:

1. Branch for Release
2. Branch for Feature
3. Branch for Team

The patterns "Branch for Feature" and "Branch for Team" are more commonly used by large project teams.

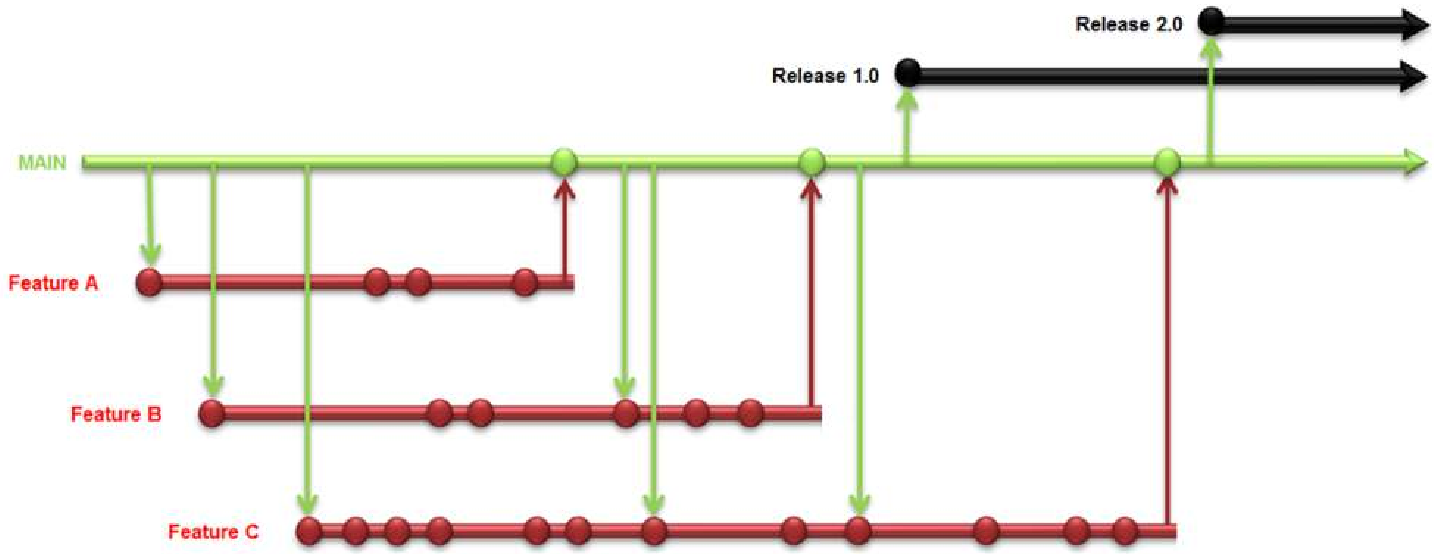
Branch for Release

- In this case, a branch is created shortly before a release. Once the branch is created, testing and validation of the release is done on the branch, while new development continues on the mainline.
- By creating a release branch, developers can keep checking it in to the mainline, while changes to the release branch are made for critical bug-fixes only.
- Features are always developed on the mainline.
- A branch is created when your code is feature-complete for a particular release and you want to start working on new features.
- Only fixes for critical defects are committed on branches, and they are merged into mainline immediately.



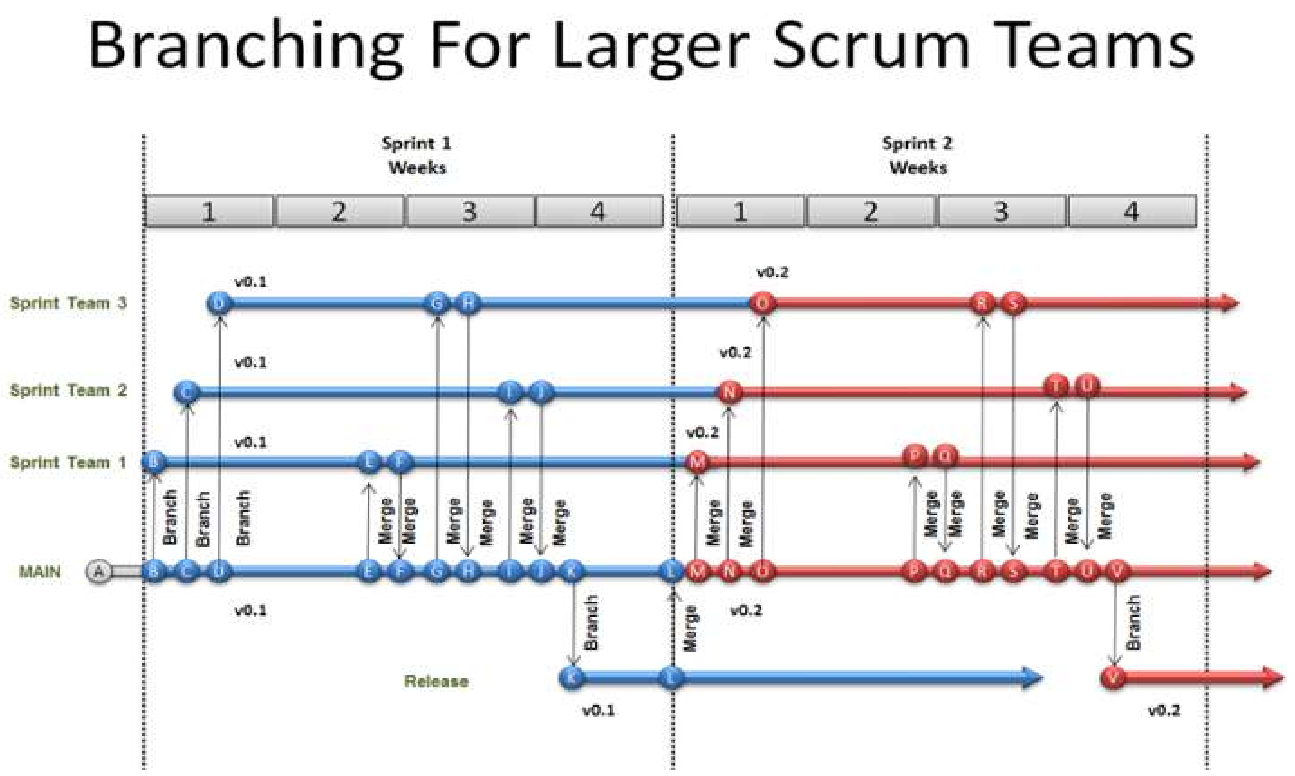
Branch for Feature

- This pattern is motivated by the desire to keep the trunk always releasable, which makes it easier for large teams to work simultaneously on features while keeping mainline in a releasable state.
- Every story or feature is developed on a separate branch. Only after a story is accepted by testers, it is merged to mainline to ensure it is always releasable.
- Any changes from mainline must be merged onto every branch on a daily basis.



Branch for Team

- As with branch by feature, the main intent here is to ensure that the trunk is always releasable.
- A branch is created for every team, and merged into the trunk only when the branch is stable.
- Every merge to any given branch should be pulled into every other branch.



Part III

Part III.
Change Management

Incident Management
Release Management
Configuration Management
Change Management

Change Management Objective

ITIL Definition

A change is the addition, modification, or removal of anything that could have an effect on IT services.

The objective of the Change Management process is to ensure that changes are deployed in a controlled way, i.e., they are evaluated, prioritized, planned, tested, and documented.

Changes must be controlled to ensure:

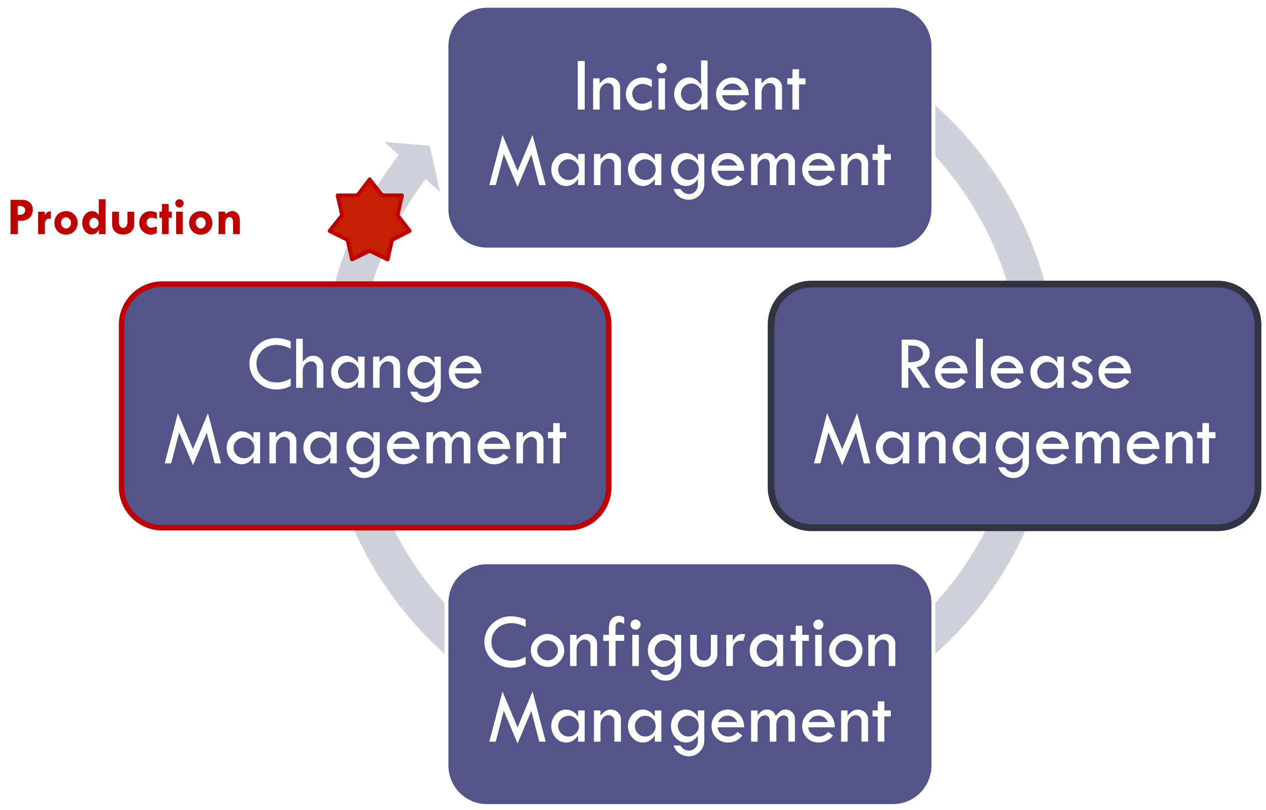
- Exposure to the risks is minimized (CHANGE = RISK);
- The severity of the impact and service interruption is minimized;
- The change is implemented successfully.

The Change Management process must:

- Use standard methods and procedures;
- Perform change impact analysis; take account of risks for the business.
- Record all changes in the CMDB (configuration management database);

Change Management vs. Release Management

Change Management controls changes implemented and delivered by the Release and Deployment Management.



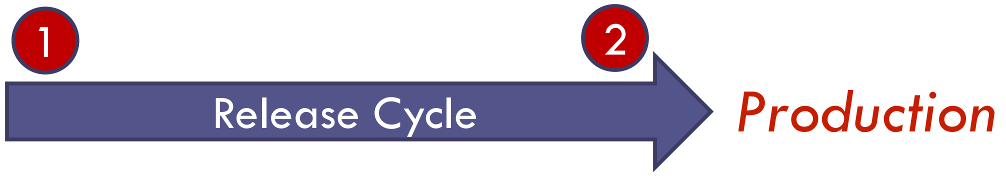
Change Manager Role

ITIL Role Definition

- The Change Manager role controls the lifecycle of all changes.
- His/her primary objective is to enable beneficial changes to be made, with minimum disruption to IT services.

Change Manager is an important role in the Change Management process that is usually established at two levels:

1. Project Team level – the role is primarily responsible for documenting, assessing, and allocating change requests (RFC "*request for change*") to releases. The Change Manager controls the entire lifecycle of a given change request from reporting to its closure. This role can be combined with other roles, for example, Project Manager, Release Manager, or Lead BA role.
2. Program/Department level – the Change Manager is a CAB Chair and is primarily responsible for controlling changes to the production environment (see next Slide).



ITIL: Change Advisory Board

Change Advisory Board (CAB) is a consultation body (group of people) that meets at fixed intervals (commonly weekly) to review submitted release tickets, scheduled for implementation this week, to approve and authorize changes to production.

CAB is usually made up of representatives from all areas within the IT organization, the business, and third parties such as suppliers.

The CAB's responsibilities include:

- Evaluating each **Change Request** ticket (and referenced RFCs) submitted by Project Managers. In particular, reviewing evidence supporting the claim that the release implementation was sufficiently tested.
- Assessing the impact of a given release on other production systems.
- Advising on the release schedule, checking it against any planned code-freeze dates in IT.
- Publishing a report on scheduled releases, their types and IT owners.

Factors in Assessing Change Request

When reviewing and assessing each Request for Change (RFC), the Change Manager should consider:

- The consequences of not making the change;
- The benefits of the change;
- The number of users affected by the change;
- The costs and risks of making the change; its impact on the existing application functionality;
- The product release cycle, code-freeze calendar, etc.

Detailed Change Impact Analysis

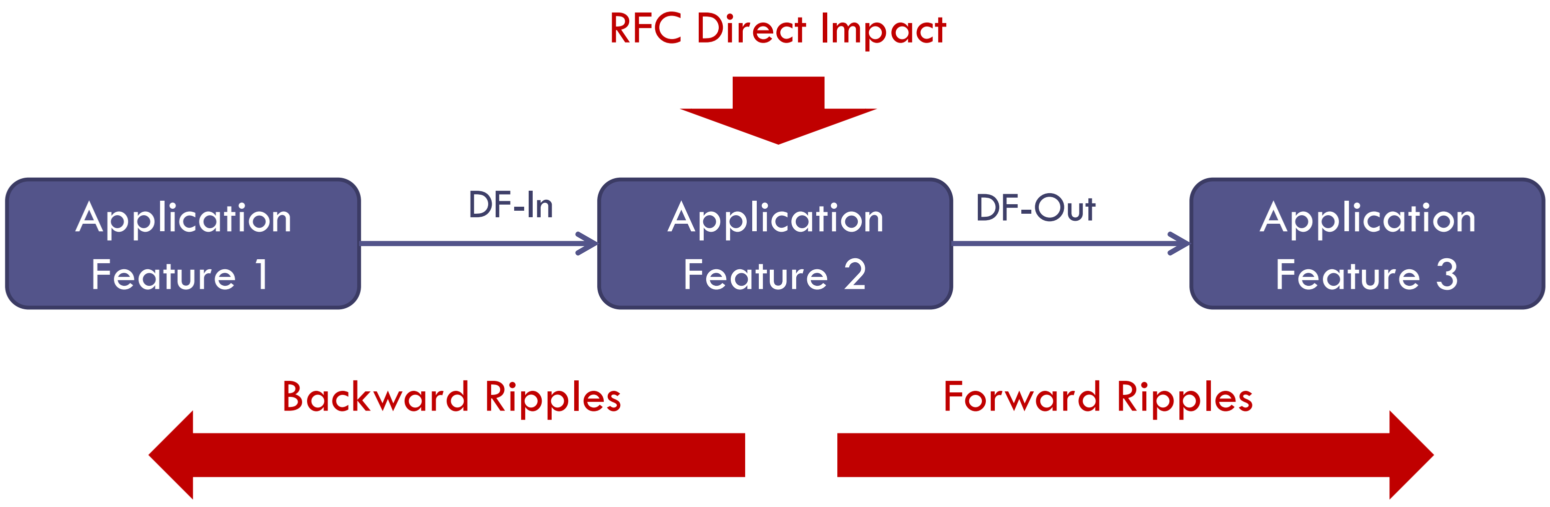
- The goal of Change Impact Analysis (CIA) is identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change.
- Business applications are evolving, driven by constantly changing business conditions. Most changes overlap with an application's functionality in production and require changing existing features, while some changes result in adding new or removing existing application features.
- Regardless of whether it is the former or latter case, the impact of changes to the existing application functionality must be sufficiently investigated and understood.
- The lack of this understanding is a common reason for production instability and incidents that can cause significant financial losses to the business.
- Performing effective CIA is very important, in particular, on agile projects where the project team lacks detailed functional specifications.

Change Impact Analysis is a critical task of the Change Management process.

Change Ripple-Effect Analysis

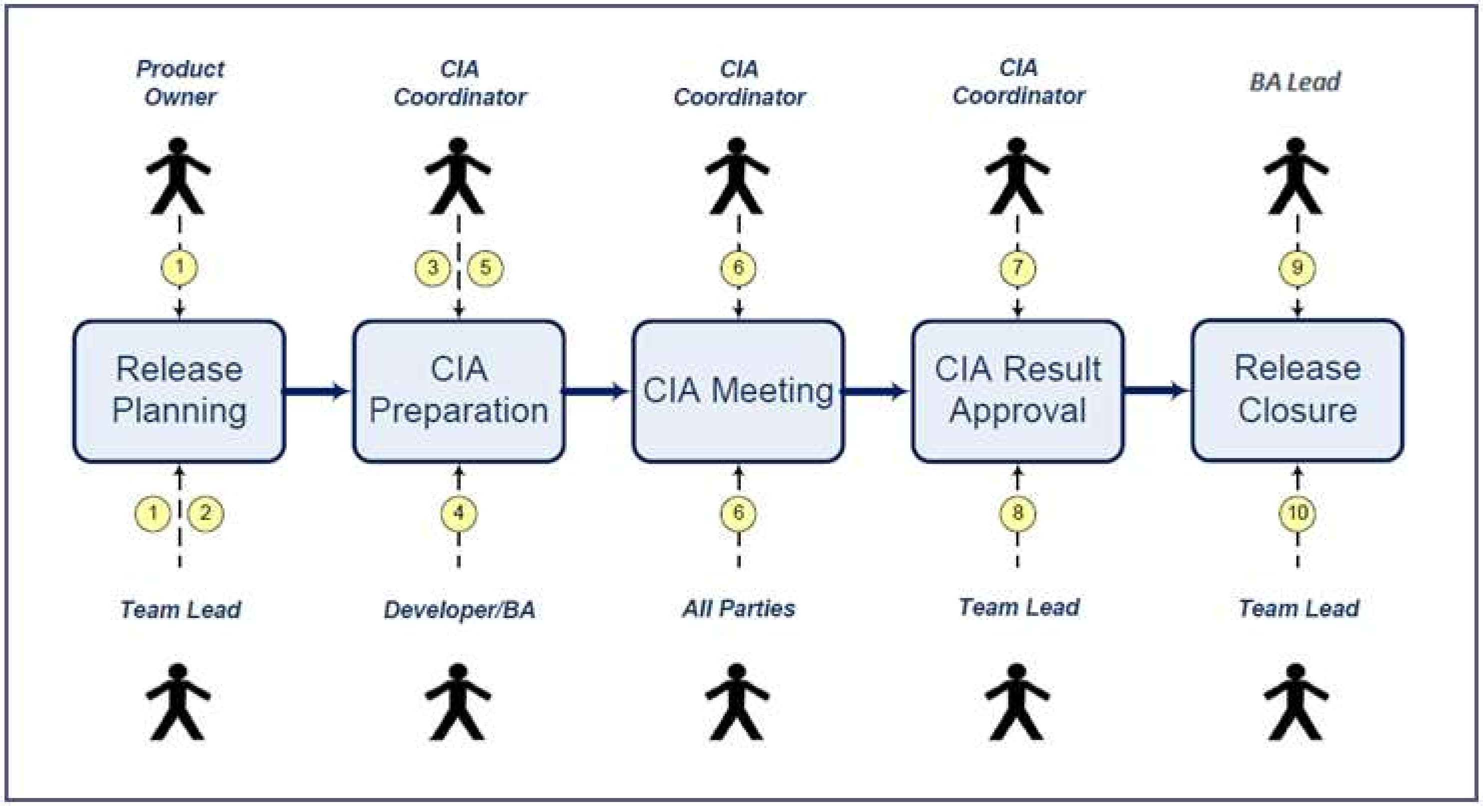
Ripple-effect analysis is a special case of change impact analysis where we investigate impact propagation based on data flows.

Backward Ripple Effect Analysis	**Forward Ripple Effect Analysis**
The investigation of backward ripples is intended to answer the question "*how does the data get here?*"	The investigation of forward ripples is intended to answer the question "*where is the data used from here?*"



CIA Process Phases and Roles

The CIA process involves multiple project roles.



See task descriptions on the next slide.

CIA Process Roles and Responsibilities

1. **Product Owner** and **Team Lead**: finalize the release scope, i.e., a list of change requests (a.k.a. business requirements).
2. **Team Lead**: assigns change requests to developers to analyze and implement.
3. **CIA Coordinator**: schedules a CIA meeting.
4. **Assigned Developers/BA**: prepare a CIA case for each change request.
5. **CIA Coordinator**: reconciles inputs from Developers, prepares the meeting materials.
6. **CIA Coordinator**: conducts a CIA meeting (Product Owner, Developers, QA parties are invited):
   a. **Developers/BA**: present a CIA case for each change request.
   b. **All Parties**: examine and validate the CIA results.
   c. **CIA Coordinator**: facilitates a discussion, takes notes.
7. **CIA Coordinator**: publishes the final CIA results after the meetings.
8. **Team Lead**: approves the CIA results for the release.
9. **BA Lead**: baselines product requirements.
10. **Team Lead**: approves the latest requirements baseline.

To make the CIA process more practical, some roles can be combined and assigned to the same team member.

Conducting a CIA Meeting

CIA meeting parties

CIA should be performed as a formal [1 hr] session where all release parties are invited – the product owner, the developers assigned to implement a release, and the QA personnel assigned to test the release.

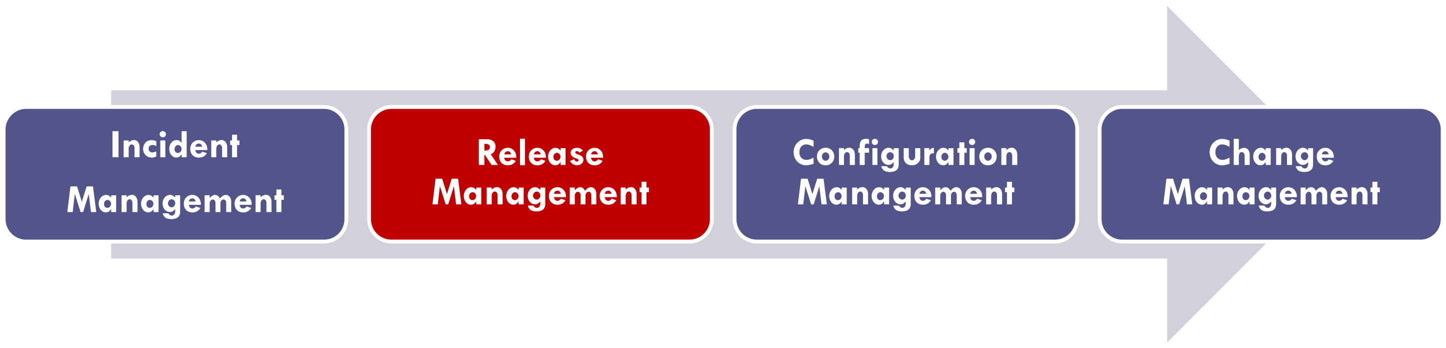
CIA meeting outcome

The CIA meeting outcome is the agreement among all release parties on the scope of release implementation and testing:

- Directly impacted features represent the scope of **functional testing.**
- Indirectly impacted features represent the scope of **regression testing.**
- Impacted system interface requirements indicate the scope of **end-to-end testing.**

Part IV

Part IV.
Release and Deployment Management



Release and Deployment Management Objective

ITIL Definitions

- A *release* is a set of new or changed configuration items that are tested and will be implemented in production together.
- *Release and Deployment Management* aims to build, test, and deliver the capability to provide the services specified by Service Design that will accomplish the stakeholders' requirements and deliver the intended objectives.

The objective of Release and Deployment Management is to ensure that:

- Release and deployment plans are in place and approved;
- Release packages are deployed successfully;
- Knowledge transfer to the customers takes place;
- There is **minimum disruption to the service**.

ITIL Types of Releases

- A system release is a version (baseline) of a software system that is distributed to customers.
- ITIL defines the following types of releases:
  - *major* releases, which deliver significant new functionality,
  - *minor* releases (a.k.a. maintenance releases), which deliver a few small improvements, e.g., repair bugs and fix customer problems that have been reported, code refactoring to improve performance, etc.
  - *emergency* releases, which deliver fixes, usually as a temporary solution, to critical production incidents.
- The process of approving releases for production deployment can be different based on a release type.
- For custom software or software product lines, releases of the system may have to be produced for individual customers. In this case the customers will be running several different versions of the system at the same time.

Specifics of Large IT Organizations

Transaction Flow

- A portfolio of business applications in large IT departments can include thousands of systems, where many of them are interconnected to support business transaction flows.
- A change implemented in one system can be propagated to and impact other upstream and downstream systems.

Segregation of Duties

- With the adoption of the Sarbanes-Oxley (SOX) Act in 2002, many IT organizations are now required to establish internal controls and implement the principle of "segregation of duties" to achieve SOX compliance.
- In this case, an IT operating model includes a number of separate groups in the production management and technology infrastructure departments that are jointly responsible for application releases and support in production.

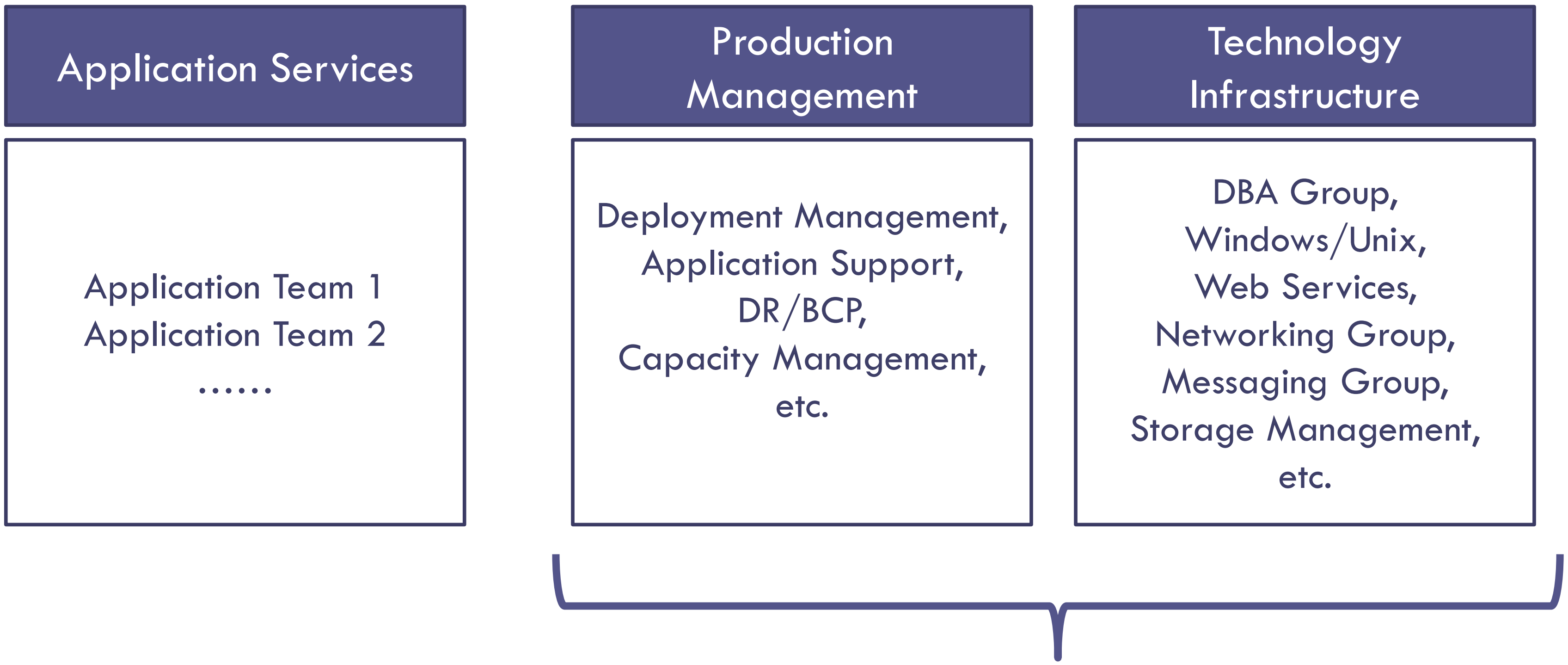
Various IT groups can be impacted by a given application release that makes it necessary to perform detailed impact analysis and planning of production releases.

An Example of an IT Department Structure

The “Segregation of Duties” principle illustrated:

No access to production

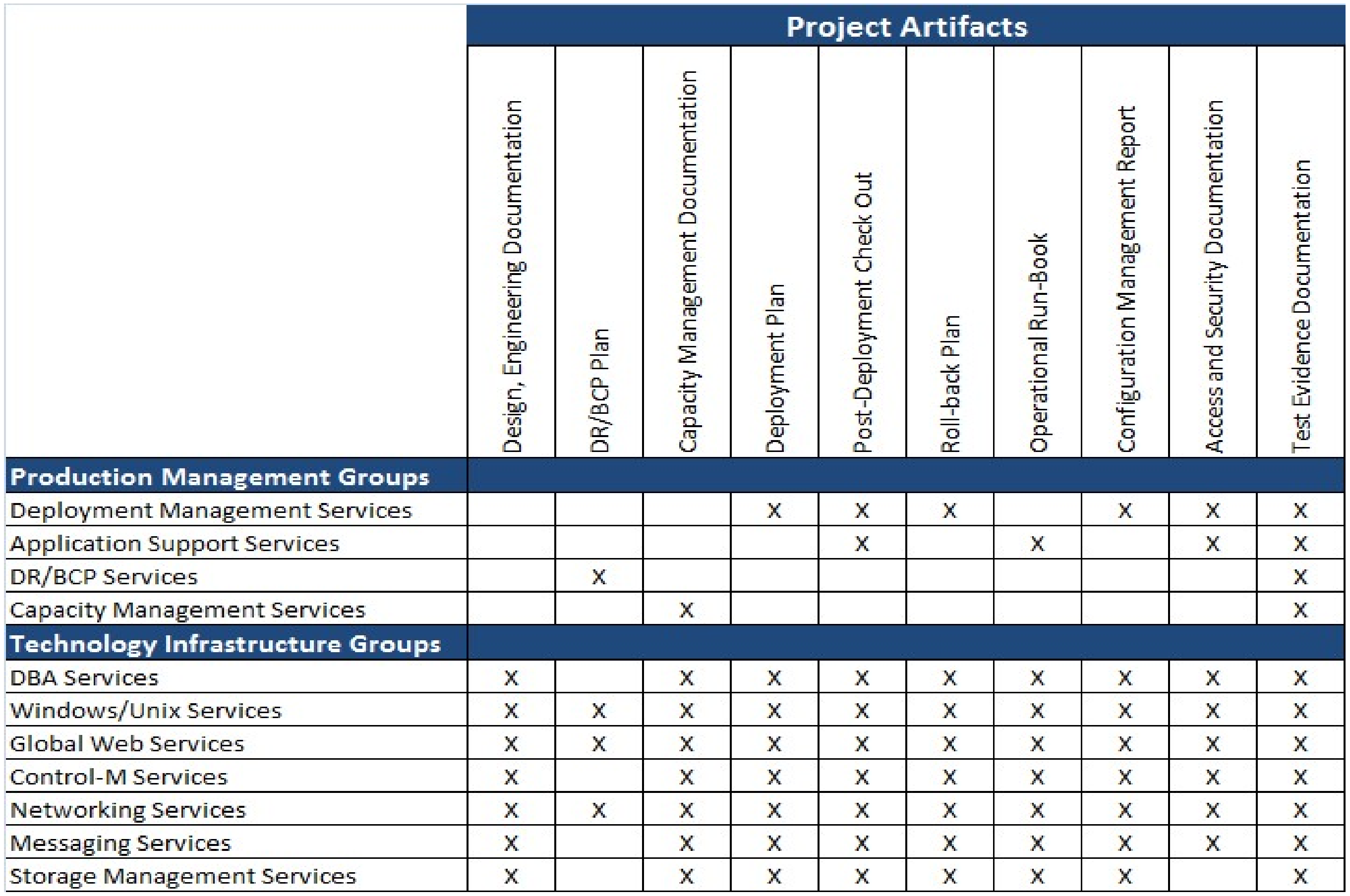
Groups responsible for production support



Operational procedures of these groups are commonly defined based on the ITIL Framework.

IT Operating Model Example

An IT Operating Model example, common to investment banks.

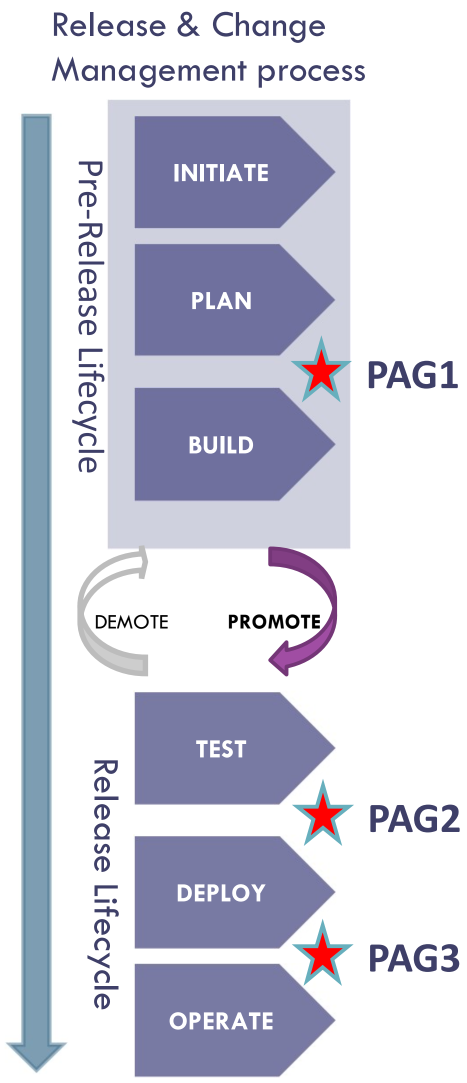


	Project Artifacts									
	Design, Engineering Documentation	DR/BCP Plan	Capacity Management Documentation	Deployment Plan	Post-Deployment Check Out	Roll-back Plan	Operational Run-Book	Configuration Management Report	Access and Security Documentation	Test Evidence Documentation
Production Management Groups										
Deployment Management Services				X	X	X		X	X	X
Application Support Services					X		X		X	X
DR/BCP Services		X								X
Capacity Management Services			X							X
Technology Infrastructure Groups										
DBA Services	X		X	X	X	X	X	X	X	X
Windows/Unix Services	X	X	X	X	X	X	X	X	X	X
Global Web Services	X	X	X	X	X	X	X	X	X	X
Control-M Services	X		X	X	X	X	X	X	X	X
Networking Services	X	X	X	X	X	X	X	X	X	X
Messaging Services	X		X	X	X	X	X	X	X	X
Storage Management Services	X		X	X	X	X	X	X		X

Groups Impacted by a Production Release

- In large IT organizations, which adopted the "segregation of duties" principle, there could be many IT groups, outside the application team, who could be impacted by and should be participating in application releases.
- For better coordination of these groups, a release process can include some control points, a.k.a., process gates (see next slide).

Production Acceptance Gates (PAG)



The **PAG Framework** is comprised of three gates that provide necessary control over the passage of releases to production.

PAG1

"Planning"

Engage testing and production support personnel to plan their services for a release.

PAG2

"Production Acceptance"

Review the evidence of application testing completion and agree that the application is ready for production deployment and release approval at a CAB session.

PAG3

"Ready for Business"

Confirm that after deployment the application is ready for business.

Release Deployment Planning

Deployment (Roll-out) – ITIL Definition

(Service Transition) The activity responsible for movement of new or changed hardware, software, documentation, process, etc. to the Live Environment. Deployment is part of the **Release and Deployment Management** process.

For market or custom software products, as well as the technical work involved in creating a release distribution, advertising and publicity material have to be prepared and marketing strategies put in place to convince customers to buy the new release of the system.

CI/CD DevOps – Deployment Strategy

- Continuous Integration (CI) and Continuous Delivery (CD), also known as CI/CD, embodies a culture, operating principles, and a set of practices that application development teams use to deliver code changes more frequently and reliably.
- Continuous integration requires that every time somebody commits any change, the entire application is built and a set of automated tests is run against it. If the test process fails, the development team fixes the problem immediately.
- The goal of continuous integration is that the software is in a working state, i.e., is ready for deployment all the time.

CI/CD Explained

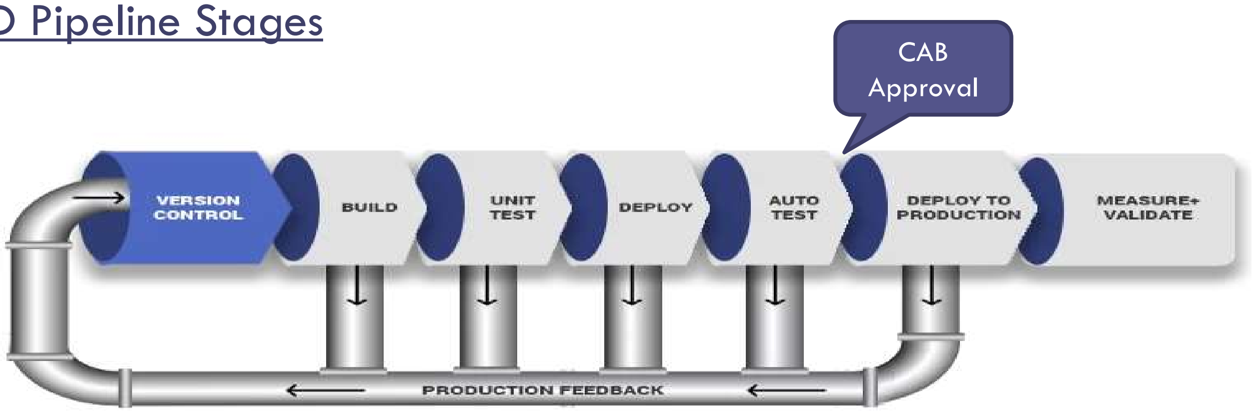
What is CI/CD?

- CI/CD is a way of developing software in which you're able to release updates at any time in a sustainable way. When changing code is routine, development cycles are more frequent, meaningful and faster.

What is the CI/CD Pipeline?

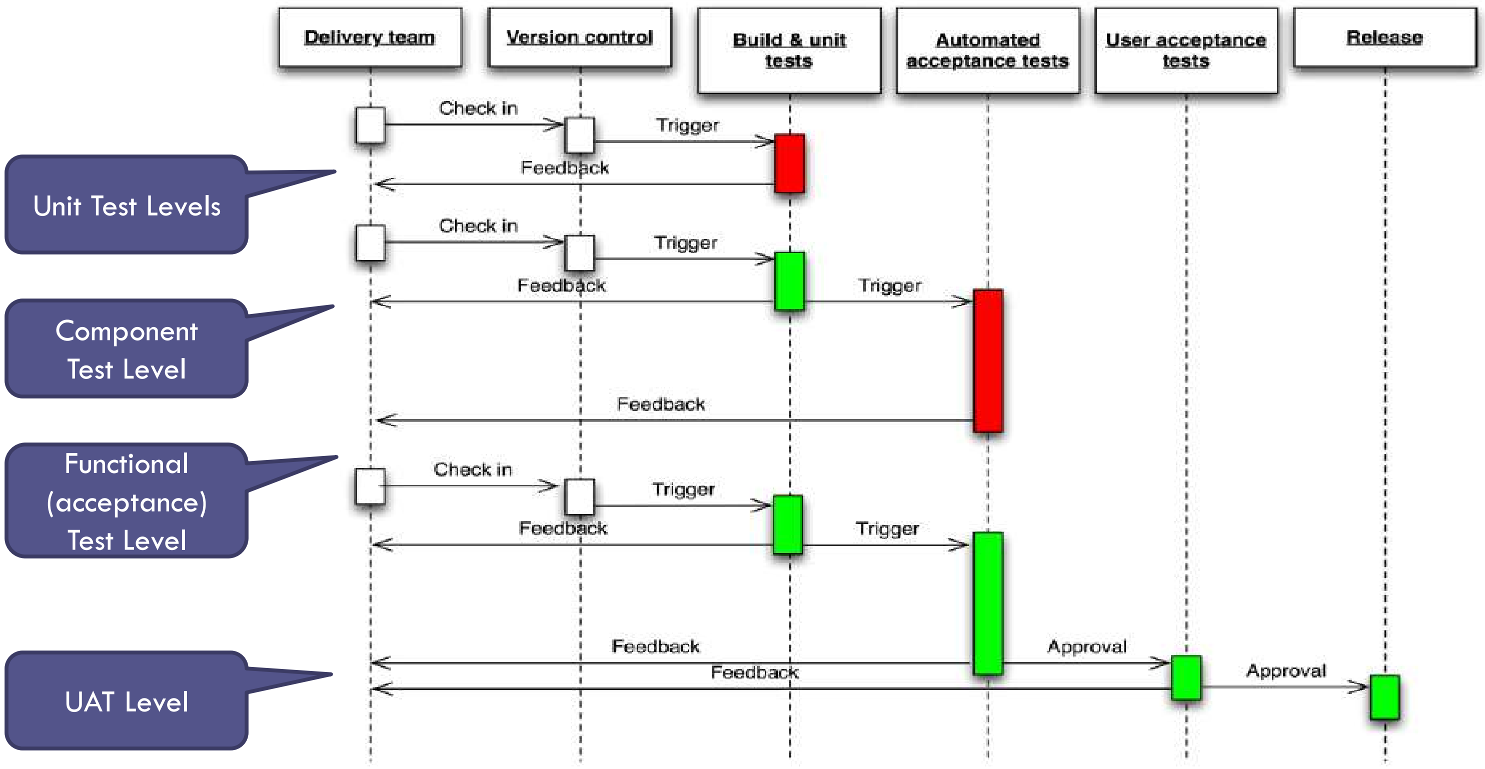
- The continuous integration/continuous delivery (CI/CD) pipeline is an agile DevOps workflow focused on a frequent and reliable software delivery process.
- A key characteristic of the CI/CD pipeline is the use of automation at every stage of the delivery process to ensure code quality.

CI/CD Pipeline Stages

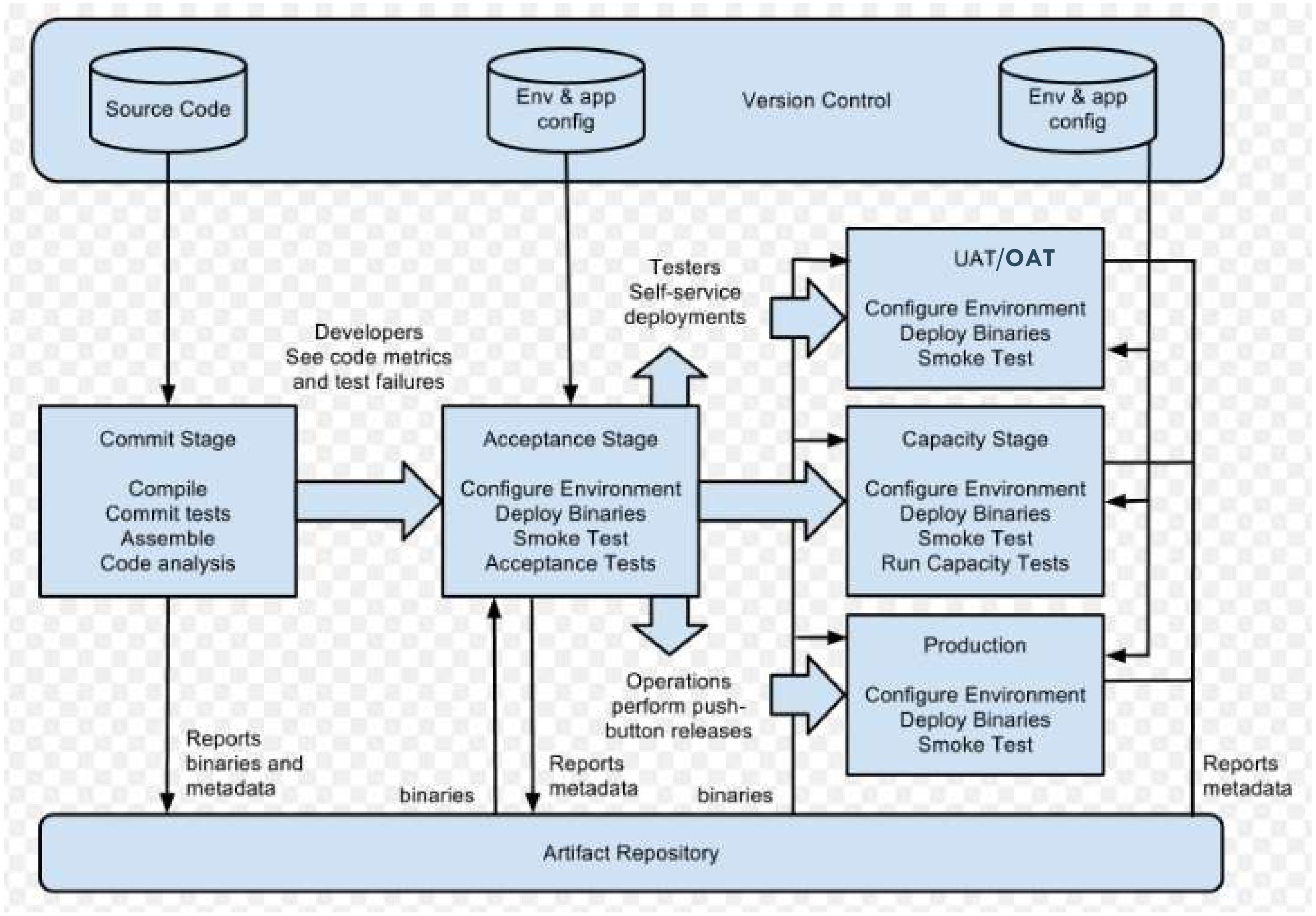


DevOps Pipeline Feedback

- The CI/CD pipeline is a continuous cycle of build, test and deploy.
- Every time code is tested, developers can quickly take action on the feedback and improve the code.
- Testing is performed and feedback received at multiple levels.



Conceptual Deployment Pipeline Flow



The diagram is explained on the next slide.

Deployment Pipeline Explained

- A ***deployment pipeline*** is a process model for software delivery that includes the following stages:
- The Commit stage asserts that the system works at the technical level, i.e., it compiles and passes a suite of automated unit tests. Here we assemble the executable code into binaries and store them in an artifact repository.
- The Automated Acceptance Test stage asserts that the system works at the functional and non-functional levels, i.e., it meets the needs of its users.
  - *After this point, the pipeline branches to enable deployments to various environments – UAT/OAT, Staging, Production environments. To facilitate this, an automated deployment script is developed and used for migrations.*
- The Manual Test (UAT, Capacity) stage asserts that the system is usable, i.e., fulfills its requirements and provides value to its users.
- The Production stage delivers the system to users.

Continuous Integration Best Practices

Continuous Integration goal is to ensure that all software is working all the time. The following practices can help achieve this goal:

1. Do not Check-in on a broken build
2. Always run all commit Tests locally before Committing
3. Wait for Commit tests to pass, before moving on
4. Never go home on a broken build
5. Always be prepared to revert to the previous revision
6. Time-box fixing before reverting
7. Don’t commit failing tests
8. Take responsibility for all Breaks that result from your code changes

Release Deployment Best Practices

1. Have a release plan that is agreed to with other stakeholders – connected application teams, CAB, production support and infrastructure teams.
2. Automate a deployment process as much as possible, minimize the chance of manual deployment mistakes.
3. Rehearse the deployment process often in production-like environments (*Operational Acceptance Testing*) , so you can debug and validate the process.
4. Have post-deployment procedures (*technical and business check-out*) to demonstrate that a released system is ready for business.
5. Have a back-out (a.k.a. roll-back) plan, the ability to back out a release if things don't go according to plan, i.e., if post-deployment check out has failed.

Popular Continuous Integration Tools

Tool Name	Description
Azure Pipelines	Azure Pipelines (part of Microsoft Azure) simplifies the creation and testing of code projects prior to making them freely accessible. It may be used with practically any programming language or type of project. Azure Pipelines lets you build and test your code concurrently while distributing it to any destination.
Bamboo	Bamboo Server helps DevOps, and CI/CD teams streamline software development and delivery using the power of automation, integrations, and workflow management. It is part of the Atlassian ecosystem of products and can be used in conjunction with your code pipelines in Bitbucket.
Jenkins	Jenkins is an open-source server for automation that is free to use. In software development, it aids in the automation of the processes associated with creating, testing, and deploying, hence helping the methods of continuous integration and delivery. In this case, it is a server-based solution that operates in servlet containers such as the Apache Tomcat web server.
GitLab	GitLab is a collection of tools intended to be used in various software development life cycle stages. At its core, it is a Git repository manager designed to be used on the web, with capabilities such as tracking issues, reporting, and an online wiki.

Key Points

- The IT Infrastructure Library (ITIL) is the most common framework that focuses on aligning IT services with the needs of business. Service Transition is one of the ITIL process areas discussed in this lecture.
- The main objective of Service Transition is to plan and manage the capacity and resources required to package, build, test and deploy a release into production and establish the service specified in the customer and stakeholder requirements.
- Configuration Management is concerned with the policies, processes, and tools for managing changing software systems.
- Configuration Management activities include configuration planning, configuration identification, configuration control, status accounting, and configuration audits.
- Configuration Management is closely related to other ITIL processes like Incident Management, Change Management, and Release Management.
- Configuration Manager and Change Control Board are two important roles of the Configuration Management process.
- Version Management is the process of keeping track of different versions of software components or configuration items and the systems in which these components are used.

Key Points (continued)

- The objective of the Change Management process is to ensure that changes are deployed in a controlled way, i.e., they are evaluated, prioritized, planned, tested, and documented. Change Management controls changes implemented and delivered by the Release and Deployment Management.
- Change Manager is an important role in the Change Management process that controls the lifecycle of all changes. This role can be established at two levels - a Program Level and Project Team level.
- Change Impact Analysis (CIA) is an important task of the Change Management process on software maintenance projects. The goal of change impact analysis is identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change.
- Release and Deployment Management aims to build, test and deliver the capability to provide the services specified by service design and that will accomplish the stakeholders' requirements and deliver the intended objectives.
- ITIL defines three types of releases - *Major* release, *Minor* release, and *Emergency* release.
- Deployment (Roll-out) is the activity responsible for movement of new or changed hardware, software, documentation, process, etc. to the Live Environment.