IT 495
Selected Topics in Information Technology-1

IT Service Analysis, Design, and Operation

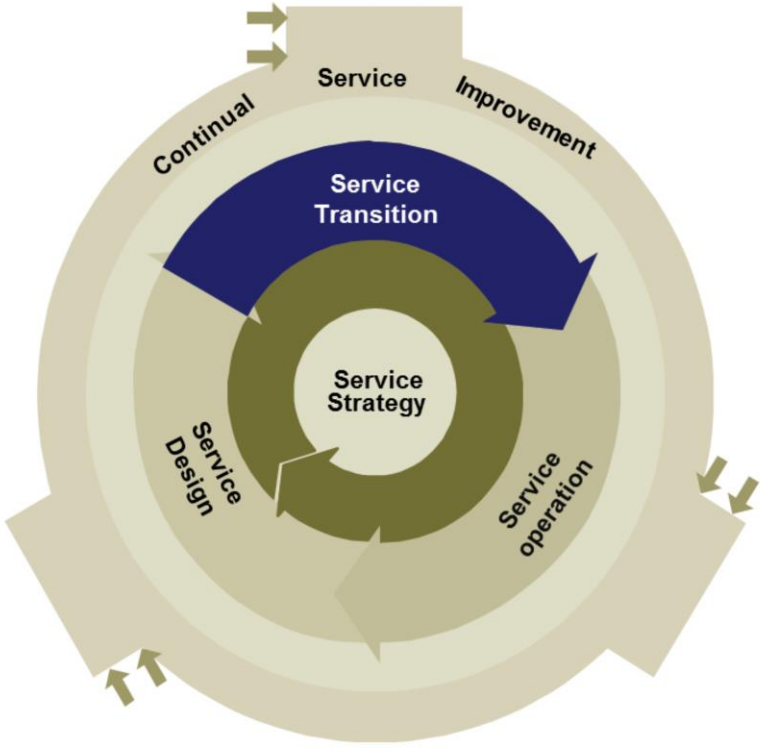
Part 4: Service Transation

Haitham S. Hamza, Ph.D.
Cairo University
Spring 2023

Acknowledgment

- Slides are based on the ITIL® Foundation Course developed by the Software Engineering Competence Center (SECC).
- All ITIL® Figures are © OGC's Official Accreditor - The APM Group Limited 2008
- I have modified them and added new slides

Service Life-Cycle



Contents

Introduction

Service Transition – Purpose and Objectives

- **Purpose:** ensure that new, modified or retired services meet the expectations of the business as documented in the service strategy and service design stages of the lifecycle

- **Objectives:**
  - Plan and manage service changes efficiently and effectively
  - Manage risks relating to new, changed or retired services
  - Successfully deploy service releases into supported environments
  - Set correct expectations on the performance and use of new or changed services
  - Ensure that service changes create the expected business value
  - Provide good-quality knowledge and information about services and service assets

The purpose of the service transition stage of the service lifecycle is to ensure that new, modified or retired services meet the expectations of the business as documented in the service strategy and service design stages of the lifecycle.

The objectives of service transition are to:

- Plan and manage service changes efficiently and effectively
- Manage risks relating to new, changed or retired services
- Successfully deploy service releases into supported environments
- Set correct expectations on the performance and use of new or changed services
- Ensure that service changes create the expected business value
- Provide good-quality knowledge and information about services and service assets.

In order to achieve these objectives, there are many things that need to happen during the service transition lifecycle stage. These include:

- Planning and managing the capacity and resources required to manage service transitions
- Implementing a rigorous framework for evaluating service capabilities and risk profiles before new or changed services are deployed
- Establishing and maintaining the integrity of service assets
- Providing efficient repeatable mechanisms for building, testing and deploying services and releases
- Ensuring that services can be managed, operated and supported in accordance with constraints specified during the service design stage of the service lifecycle.

Service Transition – Scope

- *ITIL Service Transition* provides guidance for:
  - the development and improvement of capabilities for transitioning new and changed services into supported environments, including release planning, building, testing, evaluation and deployment
  - Retiring services and transferring services between service providers
  - Ensuring that the requirements from service strategy, developed in service design, are effectively realized in service operation
  - Transitioning of changes in the service provider's service management capabilities
- Consideration is given to:
  - Managing the complexity associated with changes to services and service management processes
  - Allowing for innovation while minimizing the unintended consequences of change
  - Introducing new services, changing or decommissioning services
  - Transferring services to and from other service providers.

ITIL Service Transition provides guidance for the development and improvement of capabilities for transitioning new and changed services into supported environments, including release planning, building, testing, evaluation and deployment. The publication also considers service retirement and transfer of services between service providers. The guidance focuses on how to ensure that the requirements from service strategy, developed in service design, are effectively realized in service operation while controlling the risks of failure and subsequent disruption.

Consideration is given to:

•Managing the complexity associated with changes to services and service management processes

•Allowing for innovation while minimizing the unintended consequences of change

•Introducing new services

•Changes to existing services, e.g. expansion, reduction, change of supplier, acquisition or disposal of sections of user base or suppliers, change of requirements or skills availability

•Decommissioning and discontinuation of services, applications or other service components

•Transferring services to and from other service providers.

Guidance on transferring the control of services includes transfer in the following circumstances:

- Out to a new supplier, e.g. outsourcing
- From one supplier to another
- Back in from a supplier, e.g. insourcing
- Moving to a partnership or co-sourcing arrangement (e.g. partial outsourcing of some processes)
- Multiple suppliers, e.g. co-sourcing or multi-sourcing
- Joint venture
- Down-sizing, up-sizing (right-sizing) and off-shoring
- Merger and acquisition.
- In reality, circumstances generate a combination of several of the above options at any one time and in any one situation.

The scope also includes the transition of changes in the service provider's service management capabilities that will impact on the ways of working, the organization, people, projects and third parties involved in service management.

Service Transition – Value to Business

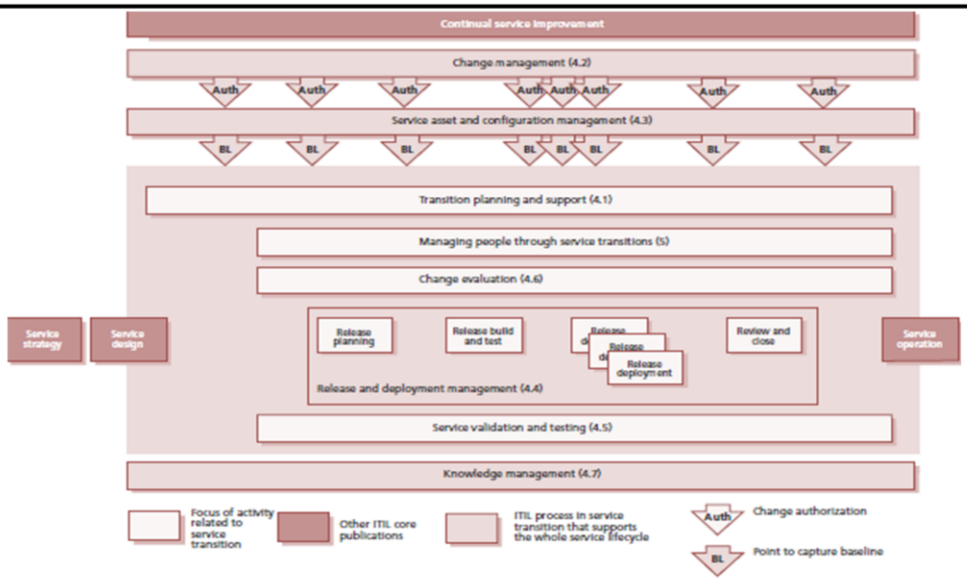
- Enable more accurate estimation for the cost, timing, resource requirement and risks associated with the ST
- Result in higher volumes of successful change
- Enable service transition assets to be shared and re-used
- Reduce delays from unexpected clashes and dependencies
- Reduce the effort spent on managing the ST test
- Improve expectation setting for all stakeholders
- Increase confidence that the new or changed service can be delivered to specification
- Ensure that new/changed services will be maintainable
- Improve control of service assets and configurations

Selecting and adopting the best practice as recommended in this publication will assist organizations in delivering significant benefits. It will help readers to set up service transition and the processes that support it, and to make effective use of those processes to facilitate the effective transitioning of new, changed or decommissioned services.

Adopting and implementing standard and consistent approaches for service transition will:

- Enable projects to estimate the cost, timing, resource requirement and risks associated with the service transition stage more accurately
- Result in higher volumes of successful change
- Be easier for people to adopt and follow
- Enable service transition assets to be shared and re-used across projects and services
- Reduce delays from unexpected clashes and dependencies – for example, if multiple projects need to use the same test environment at the same time
- Reduce the effort spent on managing the service transition test and pilot environments
- Improve expectation setting for all stakeholders involved in service transition including customers, users, suppliers, partners and projects
- Increase confidence that the new or changed service can be delivered to specification without unexpectedly affecting other services or stakeholders
- Ensure that new or changed services will be maintainable and cost-effective
- Improve control of service assets and configurations.

Service Transition – Big Picture



Service Transition Processes

- Transition planning and support
- Service asset and configuration management
- Release and deployment management
- Service validation and testing
- Change evaluation
- Knowledge management
- Change management

Service Design Processes

Change Management

Change Management – Introduction

- Changes my arise due to:
  - Improvements and enhancements (e.g., to service cost, quality, support)
  - Need to resolve errors and adopt to changing circumstances
- Changes must be managed to balance between the need for changes and the impact of changes by:
  - Optimizing risk exposure
  - Minimizing the severity of any impact and disruption
  - Increasing probability of success at the first attempt

Changes are made for a variety of reasons and in different ways – for example:

- Proactively, e.g. when organizations are seeking business benefits such as reduction in costs, improved services or increased ease and effectiveness of support
- Reactively as a means of resolving errors and adapting to changing circumstances.

Changes should be managed in order to:

- Optimize risk exposure (supporting the risk profile required by the business)
- Minimize the severity of any impact and disruption
- Achieve success at the first attempt
- Ensure that all stakeholders receive appropriate and timely communication about the change so that they are aware and ready to adopt and support the change.

Such an approach will improve the bottom line for the business by delivering early realization of benefits (or removal of risk) while saving money and time.

An appropriate response to all requests for change entails a considered approach to assessment of risk and business continuity, change impact, resource requirements, change authorization and especially to the realizable business benefit. Risk assessment should consider the risk of not implementing the change as well as any risks that the change might introduce. This considered approach is essential to maintain the required balance between the need for change and the impact of that change.

Change Management – Purpose and Objectives

- **Purpose:** control the lifecycle of all changes, enabling beneficial changes to be made with minimum disruption to IT services.

- **Objectives**:
  - Respond to the customer's changing business requirements while maximizing value and reducing incidents, disruption and re-work
  - Respond to the business and IT requests for change to align
  - Ensure that changes are recorded and evaluated
  - Ensure that all changes to configuration items are recorded in the CMS
  - Optimize overall business risk

The purpose of the change management process is to control the lifecycle of all changes, enabling beneficial changes to be made with minimum disruption to IT services.

The objectives of change management are to:

•Respond to the customer's changing business requirements while maximizing value and reducing incidents, disruption and re-work.

•Respond to the business and IT requests for change that will align the services with the business needs.

•Ensure that changes are recorded and evaluated, and that authorized changes are prioritized, planned, tested, implemented, documented and reviewed in a controlled manner.

•Ensure that all changes to configuration items are recorded in the configuration management system.

•Optimize overall business risk – it is often correct to minimize business risk, but sometimes it is appropriate to knowingly accept a risk because of the potential benefit.

Change Management – Scope

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

(ITIL Text)

- The scope of change management covers changes to all Configuration Items (CIs) across the whole service life cycle:
  - Architectures, processes, tools, metrics and documentation,
  - IT services
  - Physical assets (e.g., servers or networks)
  - Virtual assets (e.g., virtual servers or virtual storage)
  - Agreements or contracts.
  - Changes to any of the five aspects of service design
- All changes must be recorded and managed in a controlled way

All changes must be recorded and managed in a controlled way. The scope of change management covers changes to all configuration items across the whole service lifecycle, whether these CIs are physical assets such as servers or networks, virtual assets such as virtual servers or virtual storage, or other types of asset such as agreements or contracts. It also covers all changes to any of the five aspects of service design:

•Service solutions for new or changed services, including all of the functional requirements, resources and capabilities needed and agreed

•Management information systems and tools, especially the service portfolio, for the management and control of services through their lifecycle

•Technology architectures and management architectures required to provide the services

•Processes needed to design, transition, operate and improve the services

•Measurement systems, methods and metrics for the services, the architectures, their constituent components and the processes.

Each organization should define the changes that lie outside the scope of its change management process.

Typically these might include:

•Changes with significantly wider impacts than service changes, e.g. departmental organization, policies and business operations – these changes would produce RFCs to generate consequential service changes.

•Changes at an operational level such as repair to printers or other routine service components.

Types of Change Requests

- A change request is a formal communication seeking an alteration to one or more configuration items. A change can be raised using various ways:
  - 'request for change' document, service desk call, or project initiation document
- There are three different types of service change:
  - **Standard change** A pre-authorized change that is low risk, relatively common and follows a procedure or work instruction.
  - **Emergency change** A change that must be implemented as soon as possible
  - **Normal change** Any service change that is not a standard or an emergency change.

A change request is a formal communication seeking an alteration to one or more configuration items. This could take several forms, e.g. a 'request for change' document, service desk call or project initiation document. Different types of change may require different types of change request. For example, a major change may require a change proposal, which is usually created by the service portfolio management process. An organization needs to ensure that appropriate procedures and forms are available to cover the anticipated requests. Avoiding a bureaucratic approach to documenting a minor change removes some of the cultural barriers to adopting the change management process.

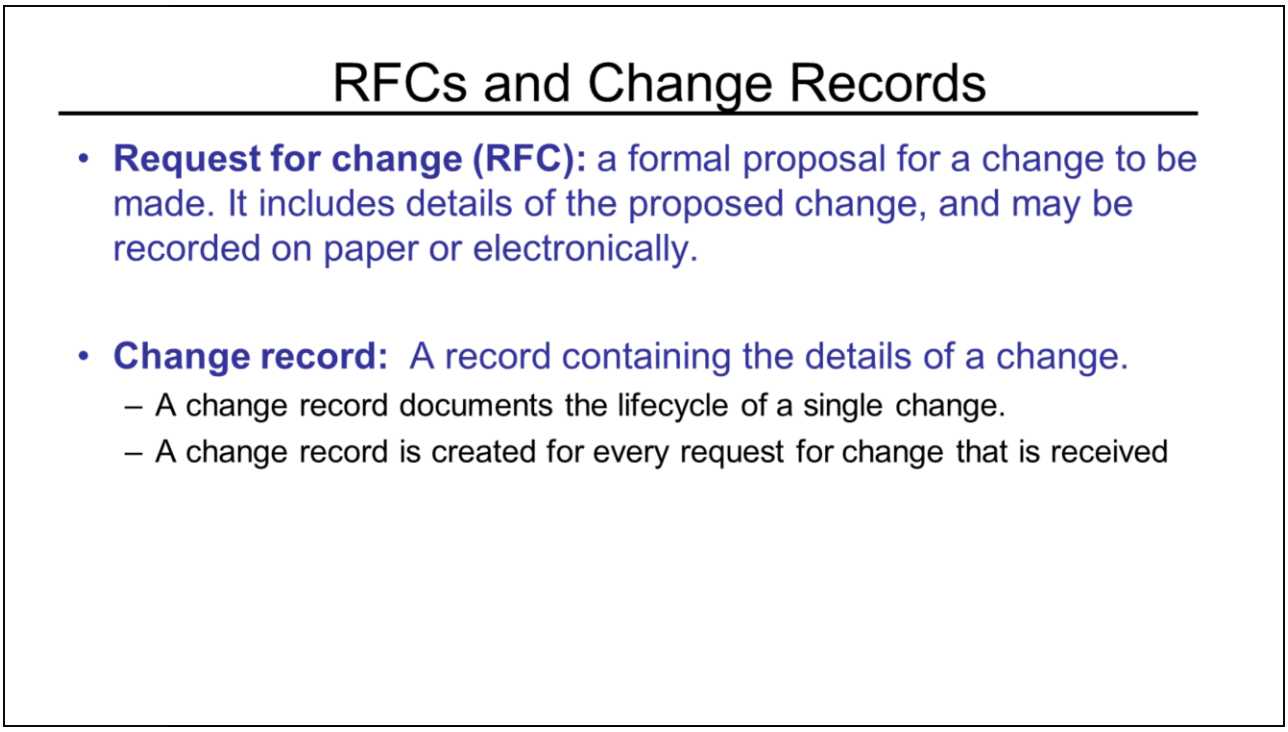
There are three different types of service change:

•**Standard change** A pre-authorized change that is low risk, relatively common and follows a procedure or work instruction.

•**Emergency change** A change that must be implemented as soon as possible, for example to resolve a major incident or implement a security patch.

•**Normal change** Any service change that is not a standard change or an emergency change.

Changes are often categorized as major, significant and minor, depending on the level of cost and risk involved, and on the scope and relationship to other changes. This categorization may be used to identify an appropriate change authority.



RFC A request for change – a formal proposal for a change to be made. It includes details of the proposed change, and may be recorded on paper or electronically. The term RFC is often misused to mean a change record, or the change itself.

Change record A record containing the details of a change. Each change record documents the lifecycle of a single change. A change record is created for every request for change that is received, even those that are subsequently rejected. Change records should reference the configuration items that are affected by the change. Change records may be stored in the configuration management system or elsewhere in the service knowledge management system.

RFCs are only used to submit requests; they are not used to communicate the decisions of change management or to document the details of the change. A change record contains all the required information about a change, including information from the RFC, and is used to manage the lifecycle of that change.

Change Models and Workflows

- A change model is a way of predefining the steps that should be taken to handle a particular type of change in an agreed way
- The change model includes:
  - Ordered steps that should be taken to handle the change
  - Responsibilities (including change authorities)
  - Timescales and thresholds for completion of the actions
  - Escalation procedures

Organizations will find it helpful to predefine change models – and apply them to appropriate changes when they occur. A change model is a way of predefining the steps that should be taken to handle a particular type of change in an agreed way. Support tools can then be used to manage the required process. This will ensure that such changes are handled in a predefined path and to predefined timescales.

Changes that require specialized handling could be treated in this way, such as:

- Emergency changes that may have different authorization and may be documented retrospectively
- Changes to mainframe software that may require specific sequences of testing and implementation
- Implementation of security patches for desktop operating systems that require specific testing and guaranteed deployment to large numbers of targets, some of which may not be online
- Service requests that may be authorized and implemented with no further involvement from change management.

The change model includes:

- Steps that should be taken to handle the change, including handling issues and unexpected events
- The chronological order in which these steps should be taken, with any dependences or co-processing defined
- Responsibilities – who should do what (including identification of those change authorities who will authorize the change and who will decide whether formal change evaluation is needed)
- Timescales and thresholds for completion of the actions
- Escalation procedures – who should be contacted and when.

models are usually input to the change management support tools; the tools then automate the handling, management, reporting and escalation of the process.

Change Proposals

- A change proposal is used to communicate a high-level description of the change
- Change proposals are normally created by the service portfolio management process and submitted to change management before chartering services in order to ensure that potential conflicts for resources or other issues are identified
- The change proposal should include:
  - A high-level description of the new, changed or retired service,
  - A full business case (risks, issues, and budget and financial expectations)
  - An outline schedule for design and implementation of the change

Major changes that involve significant cost, risk or organizational impact will usually be initiated through the service portfolio management process. Before the new or changed service is chartered it is important that the change is reviewed for its potential impact on other services, on shared resources, and on the change schedule.

Change proposals are submitted to change management before chartering new or changed services in order to ensure that potential conflicts for resources or other issues are identified. Authorization of the change proposal does not authorize implementation of the change but simply allows the service to be chartered so that service design activity can commence.

A change proposal is used to communicate a high-level description of the change. This change proposal is normally created by the service portfolio management process and is passed to change management for authorization. In some organizations, change proposals may be created by a programme management office or by individual projects.

The change proposal should include:

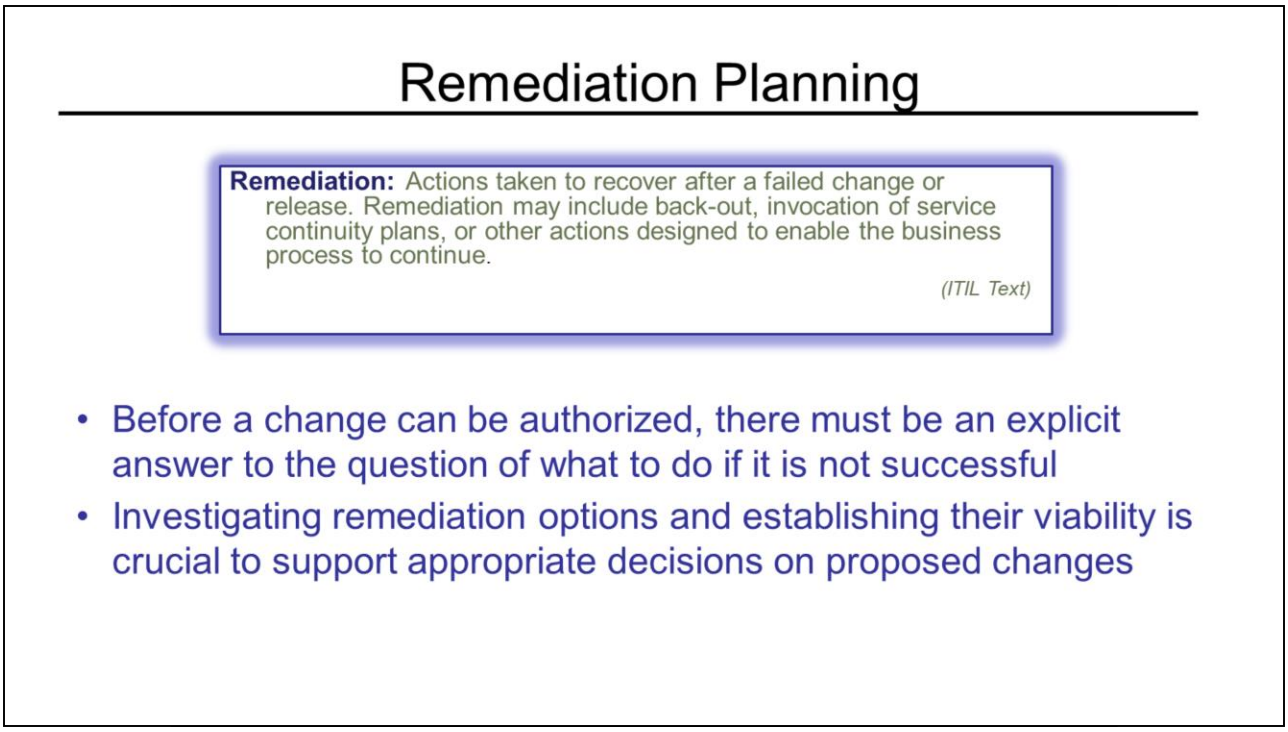
•A high-level description of the new, changed or retired service, including business outcomes to be supported, and utility and warranty to be provided

•A full business case including risks, issues and alternatives, as well as budget and financial expectations

•An outline schedule for design and implementation of the change.

•Change management reviews the change proposal and the current change schedule, identifies any potential conflicts or issues and responds to the change proposal by either authorizing it or documenting the issues that need to be resolved. When the change proposal is authorized, the change schedule is updated to include outline implementation dates for the proposed change.

After the new or changed service is chartered, RFCs will be used in the normal way to request authorization for specific changes. These RFCs will be associated with the change proposal so that change management has a view of the overall strategic intent and can prioritize and review these RFCs appropriately.



No change should be authorized without having explicitly addressed the question of what to do if it is not successful. Ideally, there will be a back-out plan, which will restore the organization to its initial state, often through the reloading of a baselined set of CIs, especially software and data. However, not all changes are reversible, in which case an alternative approach to remediation is required. This remediation may require a revisiting of the change itself in the event of failure, or may be so severe that it requires invoking the organization's business continuity plan. Only by considering what remediation options are available before instigating a change, and by establishing that the remediation is viable (e.g. it is successful when tested), can the risk of the proposed change be determined and appropriate decisions taken.

Change implementation plans should include milestones and other triggers for implementation of remediation in order to ensure that there is sufficient time in the agreed change window for back-out or other remediation when necessary.

Standard Changes – Definition

- **Standard Change is:**
  - A pre-approved change that is low Risk, relatively common and follows a Procedure or Work Instruction.
  - A change to a service or infrastructure for which the approach is pre-authorized by Change Management that has an accepted and established procedure to provide a specific change requirement.
- Examples of a standard change:
  - password reset or provision of standard equipment to a new employee.
  - upgrade of a PC in order to make use of specific standard and pre-budgeted software
- Authorization of each occurrence of a standard change ***will be granted by the delegated authority for that standard change***

A standard change is a change to a service or other configuration item for which the approach is pre-authorized by change management, and this approach follows an accepted and established procedure to provide a specific change requirement. Every standard change should have a change model that defines the steps to follow, including how the change should be logged and managed as well as how it should be implemented.

Examples might include an upgrade of a PC in order to make use of specific standard and pre-budgeted software, new starters within an organization, or a desktop move for a single user. Other examples include low impact, routine application change to handle seasonal variation.

Authorization of each occurrence of a standard change will be granted by the delegated authority for that standard change (e.g. by the budget-holding customer for installation of software from an approved list on a PC registered to their organizational unit, or by the third-party engineer for replacement of a faulty desktop printer).

Standard Changes – Properties

- Crucial elements of a standard change:
  - There is a defined trigger to initiate the RFC (if needed)
  - The tasks are well known, documented and proven
  - Authority is effectively given in advance
  - Budgetary approval will typically be preordained or within the control of the change requester
  - The risk is usually low, and always well understood.
- Typically, a change model would be associated with each standard change to ensure consistency of approach.

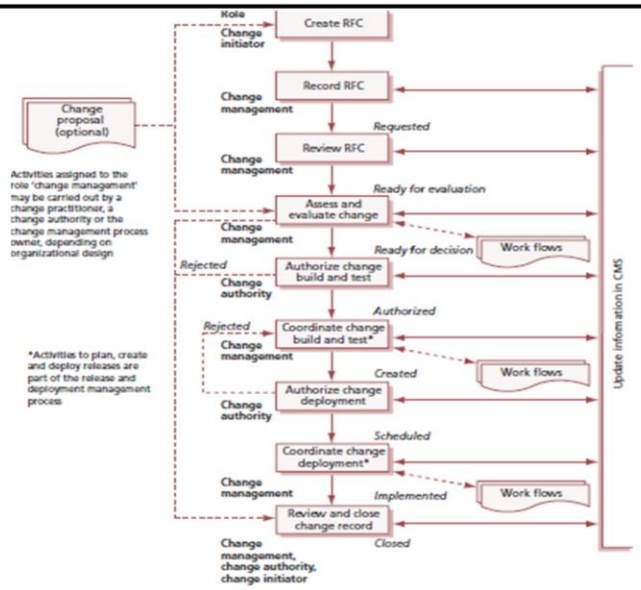
The crucial elements of a standard change are that:

- There is a defined trigger to initiate the RFC
- The tasks are well known, documented and proven
- Authority is effectively given in advance
- Budgetary approval will typically be preordained or within the control of the change requester
- The risk is usually low, and always well understood.

Once the approach to manage standard changes has been agreed, standard change processes and associated change workflows should be developed and communicated. A change model would normally be associated with each standard change to ensure consistency of approach.

Standard changes should be identified early on when building the change management process to promote efficiency. Otherwise, a change management implementation can create unnecessarily high levels of administration and resistance to the change management process.

Normal Changes – Process Flow Example



Typical activities in managing individual changes are:

- Create and record the RFC
- Review the RFC
    - Filter changes (e.g. incomplete or wrongly routed changes)
- Assess and evaluate the change
    - Establish the appropriate level of change authority
    - Establish relevant areas of interest (i.e. who should be involved in the CAB)
    - Evaluate the business justification, impact, cost, benefits, risks and predicted performance of the change
    - Submit a request for evaluation to initiate activity from the change evaluation process
- Authorize the change
    - Obtain authorization/rejection
    - Communicate the decision with all stakeholders, in particular the initiator of the request for change
- Plan updates
- Coordinate change implementation
- Review and close change
    - Collate the change documentation, e.g. baselines and evaluation reports
    - Review the change(s) and change documentation
    - Ensure that details of lessons learned have been entered into the service knowledge management system
    - Close the change document when all actions are completed.

Throughout all the process activities listed above and described within this section, information is gathered, stored in the SKMS, recorded in the CMS and reported.

Normal Changes – CAB

- **Change Advisory Board (CAB):** A body that supports the authorization of changes and assists Change Management in the assessment and prioritization of changes

- CAB should be assembled from members from both IT and business:
  - Customer(s)
  - Applications developers/maintainers
  - Specialists/technical consultants
  - Services and operations staff (e.g., service desk, test management)
  - Facilities/office services staff
  - Contractor's or third parties' representatives,

A change advisory board (CAB) is a body that exists to support the authorization of changes and to assist change management in the assessment, prioritization and scheduling of changes. A CAB is often the change authority for one or more change categories, but in some organizations the CAB just plays an advisory role. In a large organization there may be many different CABs with a global CAB that is responsible for the most significant changes and other CABs supporting different business units, geographies or technologies. It is important that each CAB has full visibility of all changes that could have an impact on the services and configuration items within its control. For each CAB meeting, members should be chosen who are capable of ensuring that all changes within the scope of the CAB are adequately assessed from both a business and a technical viewpoint.

A CAB may be asked to consider and recommend the adoption or rejection of changes appropriate for higher-level authorization, and then recommendations will be submitted to the appropriate change authority.

A practical tip worth bearing in mind is that a CAB should have stated and agreed evaluation criteria. This will assist in the change assessment activities, acting as a template or framework by which members can assess each change.

CAB needs to include people with a clear understanding across the whole range of stakeholder needs. Some of these may be permanent members of the CAB, others will be invited to participate when they are needed because of the particular changes that are being discussed. The change manager will normally chair the CAB, and potential members include:

- Customer(s)
- User manager(s)
- User group representative(s)
- Business relationship managers
- Service owners
- Applications developers/maintainers
- Specialists and/or technical consultants
- Services and operations staff, e.g. service desk, test management, IT service continuity management, information security management, capacity management
- Facilities/office services staff (where changes may affect moves/accommodation and vice versa)
- Contractors' or third parties' representatives, e.g. in outsourcing situations
- Other parties as applicable to specific circumstances (e.g. police if traffic disruptions are likely, marketing if public products could be affected).
- It is important to emphasize that a CAB:
- Will be composed of different stakeholders depending on the changes being considered
- May vary considerably in makeup, even across the range of a single meeting
- Should involve suppliers when that would be useful, for example:
- The external service provider if a significant part of the service is outsourced
- The hardware service provider when considering major firmware upgrades
- Should reflect both users' and customers' views
- Is likely to include the problem manager and service level manager and customer relations staff for at least part of the time.

Emergency Changes – Definition

- **Emergency change:** A change that must be implemented as soon as possible
  - Example: Repair an error in an IT service that is negatively impacting the business to a high degree
- Examples of changes that should not considered emergency :
  - Changes that intend to introduce immediately required business improvements (handled as ***normal changes***, assessed as having the ***highest urgency***)
  - Changes needed urgently because of poor planning or sudden changes in business requirements (treated as a ***normal change*** but given the ***highest priority***)
- Emergency change will follow the normal change procedure except that:
  - Authorization will be given by the ECAB rather than waiting for a CAB meeting
  - Testing may be reduced, or in extreme cases forgone completely, if this is considered a necessary risk to deliver the change immediately
  - Documentation, e.g. updating the change record and configuration data, may be deferred, typically until normal working hours

Emergency changes are sometimes required and should be designed carefully and tested as much as possible before use, or the impact of the emergency change may be greater than the original incident. Details of emergency changes may be documented retrospectively.

The number of emergency changes proposed should be kept to an absolute minimum, because they are generally more disruptive and prone to failure. All changes likely to be required should, in general, be foreseen and planned, bearing in mind the availability of resources to build and test the changes. Nevertheless, occasions will occur when emergency changes are essential and so procedures should be devised to deal with them quickly, without sacrificing normal management controls.

The emergency change procedure is reserved for changes intended to repair an error in an IT service that is negatively impacting the business to a high degree. Changes intended to introduce immediately required business improvements are handled as normal changes, assessed as having the highest urgency. If a change is needed urgently (because of poor planning or sudden changes in business requirements) this should be treated as a normal change but given the highest priority.

Emergency Changes – ECAB

- For emergency changes, where CAB authorization is required, this will be provided by the **emergency CAB** (ECAB)
  - Defined authorization levels will exist, and the levels of delegated authority must be clearly documented and understood
- Not all emergency changes will require the ECAB involvement; many may be predictable both in occurrence and resolution and well-understood changes available with authority delegated
- Decisions to authorize an emergency change should be documented to ensure that formal agreement from appropriate management has been received

Defined authorization levels will exist for an emergency change, and the levels of delegated authority must be clearly documented and understood. In an emergency situation it may not be possible to convene a full CAB meeting. Where CAB authorization is required, this will be provided by the emergency CAB (ECAB).

Not all emergency changes will require the ECAB involvement; many may be predictable both in occurrence and resolution and well-understood changes available with authority delegated, e.g. to service operation functions who will action, document and report on the emergency change. For example, repair or replacement of server hardware may require a small change in the server revision or configuration that can be authorized by operational staff.

It is important that any decision to authorize an emergency change is documented to ensure that formal agreement from appropriate management has been received, and to provide proper records for audits of the process.

Service Design Processes

Release and Deployment Management

Release and Deployment Management– Purpose

- plan, schedule and control the build, test and deployment of releases, and to deliver new functionality required by the business while protecting the integrity of existing services

Release and Deployment Management– Objectives

- Define and agree release and deployment management plans
- Create and test release packages
- Ensure that the integrity of a release package is maintained
- Deploy release packages from the DML to the live environment
- Ensure that all release packages can be tracked, installed, tested, verified and/or uninstalled or backed out if appropriate
- Ensure that organization and stakeholder change is managed
- Ensure that a new or changed service delivers the agreed value
- Record and manage deviations, risks and issues related to the new or changed service and take necessary corrective action
- Ensure that there is knowledge transfer to customers and users
- Ensure that skills and knowledge are transferred to service operation functions

The objectives of release and deployment management are to:

•Define and agree release and deployment management plans with customers and stakeholders

•Create and test release packages that consist of related configuration items that are compatible with each other

•Ensure that the integrity of a release package and its constituent components is maintained throughout the transition activities, and that all release packages are stored in a DML and recorded accurately in the CMS

•Deploy release packages from the DML to the live environment following an agreed plan and schedule

•Ensure that all release packages can be tracked, installed, tested, verified and/or uninstalled or backed out if appropriate

•Ensure that organization and stakeholder change is managed during release and deployment activities

•Ensure that a new or changed service and its enabling systems, technology and organization are capable of delivering the agreed utility and warranty

•Record and manage deviations, risks and issues related to the new or changed service and take necessary corrective action

•Ensure that there is knowledge transfer to enable the customers and users to optimize their use of the service to support their business activities

•Ensure that skills and knowledge are transferred to service operation functions to enable them to effectively and efficiently deliver, support and maintain the service according to required warranties and service levels.

Release and Deployment Management– Scope

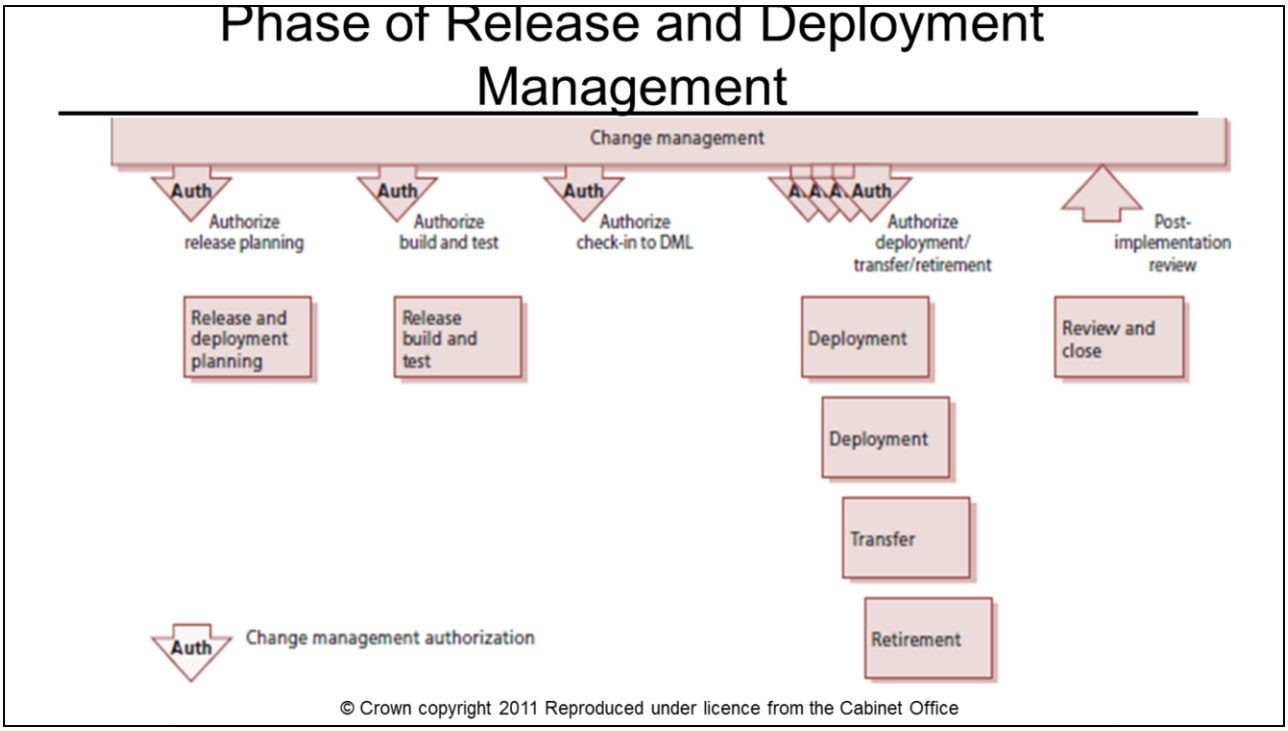
- **Scope:** includes the processes, systems and functions to package, build, test and deploy a release into live use, establish the service specified in the service design package, and formally hand the service over to the service operation functions
- Release and deployment management is responsible for ensuring that appropriate testing takes place, but the actual testing is carried out as part of the service validation and testing process
- Release and deployment management is **not** responsible for authoring changes, and requires authorization from change management at various stages in the lifecycle of a release

The scope of release and deployment management includes the processes, systems and functions to package, build, test and deploy a release into live use, establish the service specified in the service design package, and formally hand the service over to the service operation functions. The scope includes all configuration items required to implement a release, for example:

- Physical assets such as a server or network
- Virtual assets such as a virtual server or virtual storage
- Applications and software
- Training for users and IT staff
- Services, including all related contracts and agreements.

Although release and deployment management is responsible for ensuring that appropriate testing takes place, the actual testing is carried out as part of the service validation and testing process.

Release and deployment management is not responsible for authoring changes, and requires authorization from change management at various stages in the lifecycle of a release.

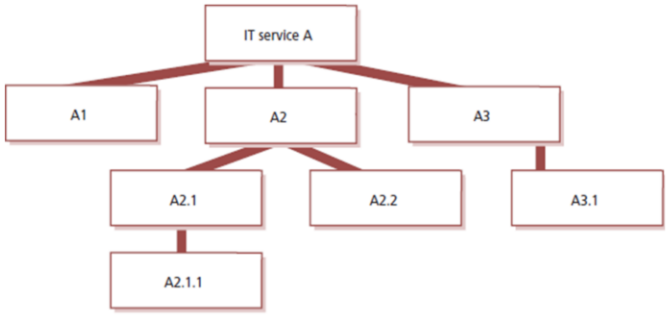


There are four phases to release and deployment management:

- **Release and deployment planning** Plans for creating and deploying the release are created. This phase starts with change management authorization to plan a release and ends with change management authorization to create the release.
- **Release build and test** The release package is built, tested and checked into the DML. This phase starts with change management authorization to build the release and ends with change management authorization for the baselined release package to be checked into the DML by service asset and configuration management. This phase only happens once for each release.
- **Deployment** The release package in the DML is deployed to the live environment. This phase starts with change management authorization to deploy the release package to one or more target environments and ends with handover to the service operation functions and early life support. There may be many separate deployment phases for each release, depending on the planned deployment options.
- **Review and close** Experience and feedback are captured, performance targets and achievements are reviewed and lessons are learned.

The figure above shows multiple points where an authorized change triggers release and deployment management activity. This does not require a separate RFC at each stage. Some organizations manage a whole release with a single change request and separate authorization at each stage for activities to continue, while other organizations require a separate RFC for each stage. Both of these approaches are acceptable; what is important is that change management authorization is received before commencing each stage.

Release Unit



A1 Server
A2 Application
A2.1 Main application, developed within the IT organization
A2.1.1 Commercial off-the-shelf reporting software used by the application
A2.2 Second application, developed within the IT organization
A3 Client software, developed within the IT organization
A3.1 Commercial off-the-shelf library providing supporting routines for the client application.

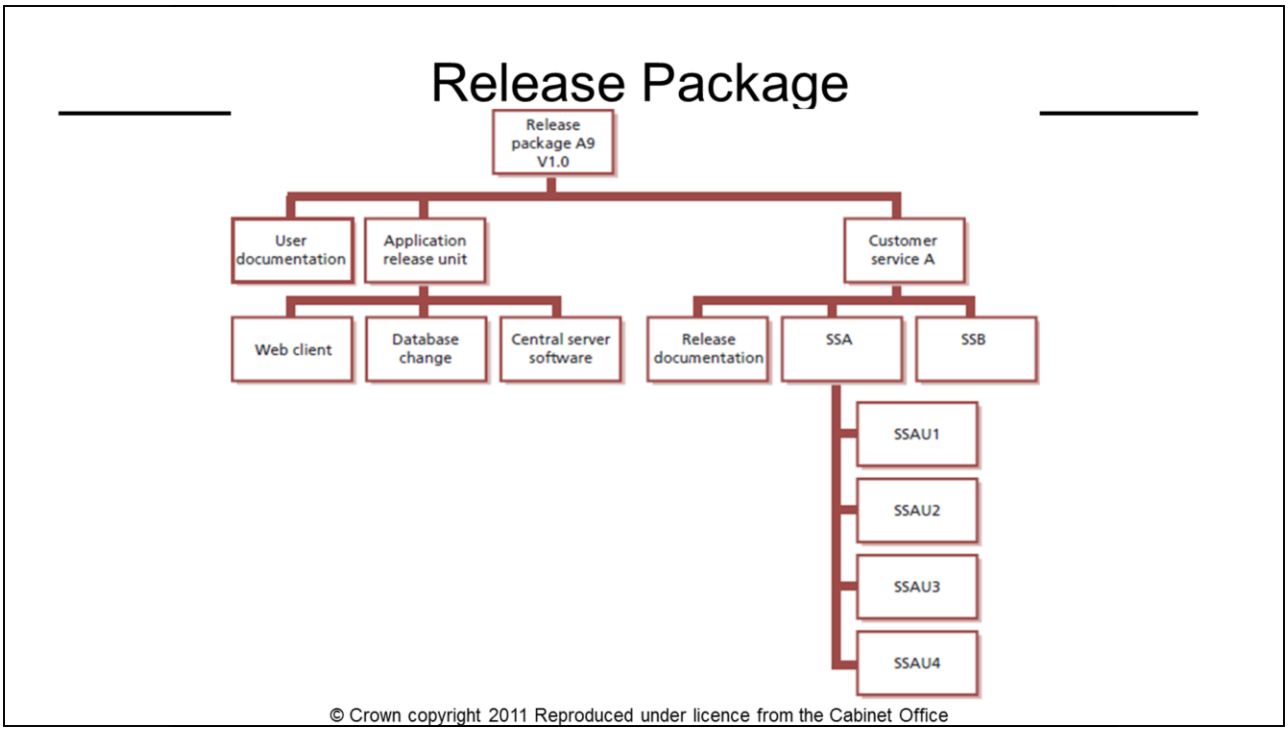
A **'release unit'** describes the portion of a service or IT infrastructure that is normally released as a single entity according to the organization's release policy. The unit may vary, depending on the type(s) or item(s) of service asset or service component such as software and hardware. The shown figure gives a simplified example showing an IT service made up of systems and service assets, which are in turn made up of service components. The actual components to be released on a specific occasion may include one or more release units, or exceptionally may include only part of a release unit. These components are grouped together into a release package for that specific release.

The general aim is to decide the most appropriate release-unit level for each service asset or component. An organization may, for example, decide that the release unit for business-critical applications is the complete application in order to ensure that testing is comprehensive. The same organization may decide that a more appropriate release unit for a website is at the page level.

Factors that should be taken into account when deciding the appropriate level for release units include:

- The ease and amount of change necessary to release a release unit
- The amount of resources and time needed to build, test, distribute and implement a release unit
- The complexity of interfaces between the proposed unit and the rest of the services and IT infrastructure
- The storage available in the build, test, distribution and live environments.

Releases should be uniquely identified according to a scheme defined in the release policy. The release identification should include a reference to the CIs that it represents and a version number that will often have two or three parts, e.g. emergency fix releases: Payroll_System v.1.1.1, v.1.1.2, v.1.1.3.



A **'release package'** is a set of configuration items that will be built, tested and deployed together as a single release. Each release will take the documented release units into account when designing the contents of the release package. It may sometimes be necessary to create a release package that contains only part of one or more release units, but this would only happen in exceptional circumstances.

A release package may be a single release unit or a structured set of release units such as the one shown in figure above. A release package may contain only part of one or more release units. This would only happen in exceptional circumstances, for example when creating an emergency release.

The example in the figure above shows an application with its user documentation and a release unit for each technology platform. On the right there is the customer service asset that is supported by two supporting services – SSA for the infrastructure service and SSB for the application service. These release units will contain information about the service, its utilities and warranties and release documentation. Often there will be different ways of designing a release package, and consideration should be given to establishing the most appropriate method for the identifiable circumstances, stakeholders and possibilities.

Where possible, release packages should be designed so that some release units can be removed if they cause issues in testing.

Release Policy Contents

- The unique identification, numbering and naming conventions for different types of release together with a description
- The roles and responsibilities at each stage in the release and deployment management process
- The requirement to only use software assets from the definitive media library
- The expected frequency for each type of release
- The approach for accepting and grouping changes into a release
- The mechanism to automate the build, installation and release distribution processes
- How the configuration baseline for the release is captured and verified against the actual release contents
- Exit and entry criteria and authority for acceptance of the release into each service transition stage and into the controlled test, training, disaster recovery and other supported environments
- Criteria and authorization to exit early life support and handover to the service operation functions

The release policy should be defined for one or more services and include:

•The unique identification, numbering and naming conventions for different types of release together with a description

•The roles and responsibilities at each stage in the release and deployment management process

•The requirement to only use software assets from the definitive media library

•The expected frequency for each type of release

•The approach for accepting and grouping changes into a release, e.g. how enhancements are prioritized for inclusion

•The mechanism to automate the build, installation and release distribution processes to improve re-use, repeatability and efficiency

•How the configuration baseline for the release is captured and verified against the actual release contents, e.g. hardware, software, documentation and knowledge

•Exit and entry criteria and authority for acceptance of the release into each service transition stage and into the controlled test, training, disaster recovery and other supported environments

•Criteria and authorization to exit early life support and handover to the service operation functions.

A release that consists of many different types of service assets may involve many people, often from different organizations. The typical responsibilities for handover and acceptance of a release should be defined and then they can be modified as required for specific transitions. The main roles and responsibilities at points of handover should be defined to ensure that everyone understands their role and level of authority and those of others involved in the release and deployment management process.

All releases should have a unique identifier that can be used by service asset and configuration management and the documentation standards.

Release Types

- **Major releases:** contain large areas of new functionality, some of which may eliminate temporary fixes to problems
- **Minor releases:** contain small enhancements and fixes, some of which may already have been issued as emergency fixes
- **Emergency releases:** contain corrections to a small number of known errors, or sometimes an enhancement to meet a high-priority business requirement

The types of release should be defined, as this helps to set customer and stakeholder expectations about the planned releases. A typical example is:

•**Major releases** Normally contain large areas of new functionality, some of which may eliminate temporary fixes to problems. A major upgrade or release usually supersedes all preceding minor upgrades, releases and emergency fixes.

•**Minor releases** Normally contain small enhancements and fixes, some of which may already have been issued as emergency fixes. A minor upgrade or release usually supersedes all preceding emergency fixes.

•**Emergency releases** Normally contain corrections to a small number of known errors, or sometimes an enhancement to meet a high-priority business requirement.

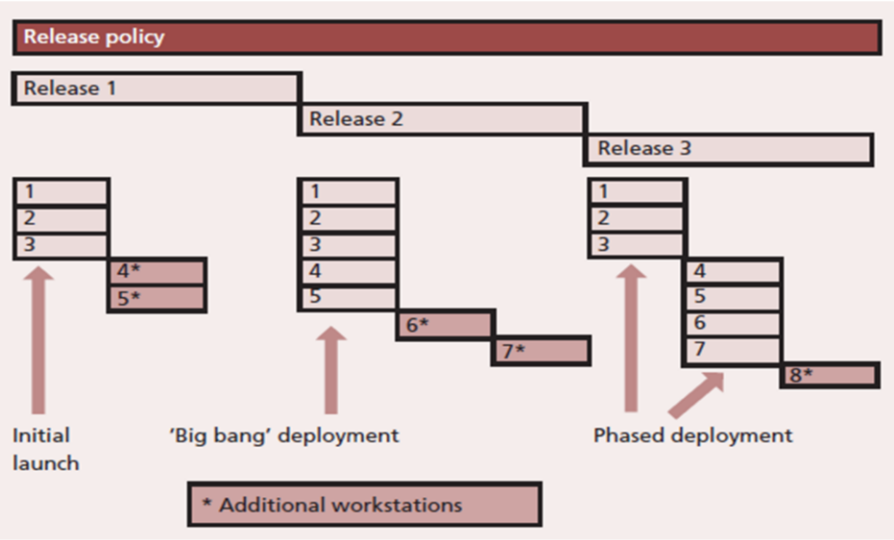
Deployments Options and Considerations

- During Service Design, the approach to transition from the current service to the new /changed service is defined.
- In particular, the SDP defines the service and solution design components to be transitioned to deliver the required service package(s) and service level package(s).
- The common options for release and development that are considered during service design are:
  - *Big Bang* versus *Phased*
  - *Push* versus *Pull*
  - *Automation* versus *Manual*

Service Design will define the approach to transitioning from the current service to the new or changed service or service offering. The SDP defines the service and solution design components to be transitioned to deliver the required service package(s) and service level package(s).

The selected option will have a significant impact on the release and deployment resources as well as the business outcomes. It is important to understand the patterns of business activity (PBA) and user profiles when planning and designing the releases.

Bing Bang vs. Phased



Options for deploying new releases to multiple locations are illustrated in Figure 4.16 and described below:

- 'Big bang' option – the new or changed service is deployed to all user areas in one operation. This will often be used when introducing an application change and consistency of service across the organization is considered important.
- Phased approach – the service is deployed to a part of the user base initially, and then this operation is repeated for subsequent parts of the user base via a scheduled rollout plan. This will be the case in many scenarios such as in retail organizations for new services being introduced into the stores' environment in manageable phases.

The shown figure illustrates a possible sequence of events over time as follows:

- There is an initial launch of the 'Release 1' of the system to three workstations (1–3).
- Two further workstations (4+5) are then added at the same time.
- 'Release 2' of the system is then rolled out in a 'big bang' approach to all workstations (1–5) at once.
- Two further workstations (6+7) are then added, in another step.
- There is a phased implementation of the upgrade to 'Release 3' of the system, initially upgrading only three workstations (1–3) and then the remaining four (4–7).
- A further workstation (8) is then added to the system.

Variations of the phased approach include:

- Portions of the service are delivered to the live environment in phases, but all end users are affected simultaneously (e.g. incremental changes to a shared application).
- Each release is deployed gradually across the total population of end users (e.g. one geographical location at a time).
- Different types of service element are deployed in separate phases, e.g. hardware changes are first, followed by user training and then by the new or changed software.
- A combination of all of these approaches is usually adopted, and the plans may deliberately allow for variations in the light of actual deployment experience.

Push vs. Pull

- **Push approach:** is used where the service component is deployed from the centre and pushed out to the target locations.
  - Delivering updated service components to all users – either in big-bang or phased form – constitutes 'push'.
- **Pull approach:** is used for software releases where the software is made available in a central location but users are free to pull the software down to their own location at a time of their choosing or when a user workstation restarts.
  - Example: virus signature updates, which are typically pulled down to update PCs and servers when it best suits the customer.

A push approach is used where the service component is deployed from the centre and pushed out to the target locations. In terms of service deployment, delivering updated service components to all users – either in big-bang or phased form – constitutes 'push', since the new or changed service is delivered into the users' environment at a time not of their choosing.

A pull approach is used for software releases where the software is made available in a central location but users are free to pull the software down to their own location at a time of their choosing or when a user workstation restarts. The use of 'pull' updating a release over the internet has made this concept significantly more pervasive. A good example is virus signature updates, which are typically pulled down to update PCs and servers when it best suits the customer; however at times of extreme virus risk this may be overridden by a release that is pushed to all known users.

In order to deploy via 'push' approach, the data on all user locations must be available. Pull approaches do not rest so heavily on accurate configuration data and they can trigger an update to user records. This may be through new users appearing and requesting downloads or expected users not doing so, triggering investigation into their continued existence. As some users will never 'pull' a release it may be appropriate to allow a 'pull' within a specified time limit and if this is exceeded a push will be forced, e.g. for an anti-virus update.

Automatic vs. Manual

- **Automatic approach:** the release and deployment activities are automatically performed.
  - Example: automated installation, builds, etc.
  - Automation will help to ensure repeatability and consistency.
  - Caution: time required to provide a well-designed and efficient automated mechanism may not always be available or viable.
- **Manual approach:** the release and deployment activities are manually performed.
  - With manual mechanisms, it is important to monitor and measure the impact of many repeated manual activities as they are likely to be inefficient and error-prone.
  - Caution: too many manual activities will slow down the release team and create resource or capacity issues that affect the service levels.

Whether by automation or other means, the mechanisms to release and deploy the correctly configured service components should be established in the release design phase and tested in the build and test stages.

Automation will help to ensure repeatability and consistency. The time required to provide a well-designed and efficient automated mechanism may not always be available or viable. If a manual mechanism is used it is important to monitor and measure the impact of many repeated manual activities as they are likely to be inefficient and error-prone. Too many manual activities will slow down the release team and create resource or capacity issues that affect the service levels.

Many of the release and deployment activities are capable of a degree of automation. For example:

- Discovery tools aid release planning.
- Discovery and installation software can check whether the required prerequisites and co-requisites are in place before installation of new or changed software components.
- Automated builds can significantly reduce build and recovery times that in turn can resolve scheduling conflicts and delays.
- Automated configuration baseline procedures save time and reduce errors in capturing the status of configurations and releases during build, test and deployment.
- Automatic comparisons of the actual ‘live’ configuration with the expected configuration or CMS help to identify issues at the earliest opportunity that could cause incidents and delays during deployment.
- Automated processes to load and update data to the CMS help to ensure the records are accurate and complete.
- Installation procedures automatically update user and licence information in the CMS.

Service Design Processes

Service Asset and Configuration Management (SACM)

SACM – Purpose and Objectives

- **Purpose:** ensure that the assets required to deliver services are properly controlled, and that accurate and reliable information about those assets is available when and where it is needed. This information includes details of how the assets have been configured and the relationships between assets

- **Objectives:**
  - Ensure that assets under the control of the IT organization are identified, controlled and properly cared for throughout their lifecycle
  - Identify, control, record, report, audit and verify services and other configuration items (CIs)
  - Account for, manage and protect the integrity of CIs through the service lifecycle
  - Ensure the integrity of CIs and configurations required to control the services
  - Maintain accurate configuration information on the historical, planned and current state of services and other CIs
  - Support efficient and effective service management processes

The purpose of the SACM process is to ensure that the assets required to deliver services are properly controlled, and that accurate and reliable information about those assets is available when and where it is needed. This information includes details of how the assets have been configured and the relationships between assets.

The objectives of SACM are to:

•Ensure that assets under the control of the IT organization are identified, controlled and properly cared for throughout their lifecycle.

•Identify, control, record, report, audit and verify services and other configuration items (CIs), including versions, baselines, constituent components, their attributes and relationships.

•Account for, manage and protect the integrity of CIs through the service lifecycle by working with change management to ensure that only authorized components are used and only authorized changes are made.

•Ensure the integrity of CIs and configurations required to control the services by establishing and maintaining an accurate and complete configuration management system (CMS).

•Maintain accurate configuration information on the historical, planned and current state of services and other CIs.

•Support efficient and effective service management processes by providing accurate configuration information to enable people to make decisions at the right time – for example to authorize changes and releases, or to resolve incidents and problems.

SACM – Scope

- The scope of SACM includes management of the complete lifecycle of every CI by:
  - Ensuring that CIs are identified, baselined and maintained
  - Ensuring that changes to them are controlled
  - Ensuring that releases into controlled environments and operational use are done on the basis of formal authorization
  - Providing configuration model of the services and service assets
- SACM may cover non-IT assets, work products used to develop the services and CIs required to support the service that would not be classified as assets by other parts of the business

The scope of SACM includes management of the complete lifecycle of every CI.

SACM ensures that CIs are identified, baselined and maintained and that changes to them are controlled. It also ensures that releases into controlled environments and operational use are done on the basis of formal authorization. It provides a configuration model of the services and service assets by recording the relationships between configuration items. SACM may cover non-IT assets, work products used to develop the services and CIs required to support the service that would not be classified as assets by other parts of the business.

The scope includes interfaces to internal and external service providers where there are assets and configuration items that need to be controlled, e.g. shared assets.

Most organizations have a process that tracks and reports the value and ownership of fixed assets throughout their lifecycle. This process is usually called *fixed asset management* or *financial asset management*. Fixed asset management maintains an asset register, which records financial information about all of the organization's fixed assets. Fixed asset management is not usually under the control of the same business unit as the IT services, but the SACM process must provide proper care for the fixed assets under the control of IT, and there must be well-defined interfaces between SACM and fixed asset management. Data from the asset register may be integrated with the configuration management system to provide a more complete view of the CIs.

SACM – Basic Definitions

- A **service asset:** is any resource or capability that could contribute to the delivery of a service.
- A **configuration item** (CI): is a service asset that needs to be managed in order to deliver an IT service.
  - All CIs are service assets, but many service assets are not configuration items.
- A **configuration record:** is a set of attributes and relationships about a CI.
  - Configuration records are stored in a configuration management database (CMDB) and managed with a configuration management system (CMS)
  - CIs are not stored in a CMDB; configuration records describe CIs that are stored in the CMDB
- The **service knowledge management system** (SKMS): is a set of tools and databases that are used to manage knowledge, information and data.
  - The SACM process is not responsible for managing the SKMS

•A **service asset** is any resource or capability that could contribute to the delivery of a service. Examples of service assets include a virtual server, a physical server, a software licence, a piece of information stored in a service management system, or some knowledge in the head of a senior manager.

•A **configuration item** (CI) is a service asset that needs to be managed in order to deliver an IT service. All CIs are service assets, but many service assets are not configuration items. Examples of configuration items are a server or a software licence, Every CI must be under the control of change management.

•A **configuration record** is a set of attributes and relationships about a CI. Configuration records are stored in a configuration management database (CMDB) and managed with a configuration management system (CMS). It is important to note that CIs are not stored in a CMDB; configuration records describe CIs that are stored in the CMDB.

•The **service knowledge management system** (SKMS) is a set of tools and databases that are used to manage knowledge, information and data. Many configuration items are available in the form of knowledge or information, and these are typically stored in the SKMS – for example, a service level agreement, a report template or a definitive media library. The SACM process is not responsible for managing the SKMS. Some items in the SKMS will be owned and managed by the SACM process, but others will be owned and managed by other processes or people.

Configuration Item (CI)

- A **configuration item (CI)** is a service asset that needs to be managed in order to deliver an IT service

- Examples of possible CIs include:
  - **Service lifecycle CIs** such as the business case, service management plans, service lifecycle plans, service design package, etc.
  - **Service CIs**:
    - Service capability assets: management, organization, processes, knowledge, people
    - Service resource assets: financial capital, systems, applications, information, data, infrastructure and facilities
    - Service model, Service package, Release package, Service acceptance criteria
  - **Organization CIs** – some documentation will be a CI in its own right and need to be controlled
  - **Internal CIs** comprising those delivered by individual projects, including tangible and intangible assets
  - **External CIs** such as external customer requirements and agreements, releases from suppliers or sub-contractors
  - **Interface CIs** that are required to deliver the end-to-end service across a service provider interface (SPI)

A configuration item (CI) is a service asset that needs to be managed in order to deliver an IT service. Configuration items may vary widely in complexity, size and type, ranging from an entire service or system including all hardware, software, documentation and support staff to a single software module or a minor hardware component. Configuration items may be grouped and managed together: e.g. a set of components may be grouped into a release. Configuration items should be selected using established selection criteria, grouped, classified and identified in such a way that they are manageable and traceable throughout the service lifecycle.

Every organization will decide whether each of the following is a configuration item or simply an attribute of a configuration item (or even something that they don't need to manage):

•**Service lifecycle CIs** such as the business case, service management plans, service lifecycle plans, service design package, release and change plans and test plans. They provide a picture of the service provider's services, how these services will be delivered, what benefits are expected, at what cost and when they will be realized.

•**Service CIs**:

•Service capability assets: management, organization, processes, knowledge, people

•Service resource assets: financial capital, systems, applications, information, data, infrastructure and facilities, financial capital, people, Service model, Service package

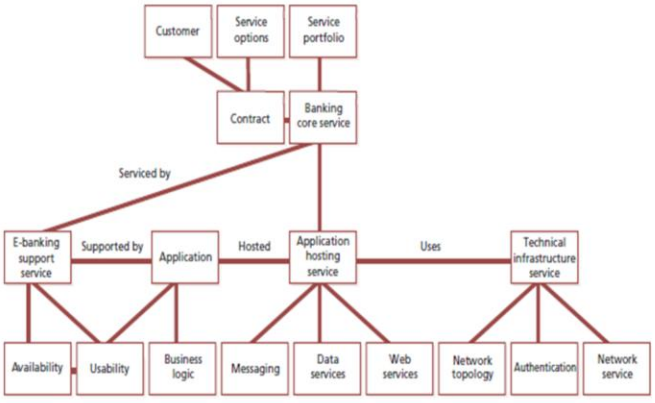
•**Organization CIs** – some documentation will define the characteristics of a CI whereas other documentation will be a CI in its own right and need to be controlled, e.g. the organization's business strategy or other policies that are internal to the organization but independent of the service provider. Regulatory or statutory requirements also form external products that need to be tracked, as do products shared among more than one group.

•**Internal CIs** comprising those delivered by individual projects, including tangible (data centre) and intangible assets such as software that are required to deliver and maintain the service and infrastructure.

•**External CIs** such as external customer requirements and agreements, releases from suppliers or sub-contractors and external services.

•**Interface CIs** that are required to deliver the end-to-end service across a service provider interface (SPI), for example an escalation document that specifies how incidents will be transferred between two service providers.

Configuration Model



- The configuration model records the relationships between CIs for the services and infrastructure
- Configuration Management's logical is THE model the represents a single common representation that can be used by all parts of IT Service Management, and beyond, such as HR, finance, supplier and customers

Service asset and configuration management delivers a model of the services, assets and the infrastructure by recording the relationships between configuration items as shown in figure above. This enables other processes to access valuable information, for example:

- To assess the impact and cause of incidents and problems
- To assess the impact of proposed changes
- To plan and design new or changed services
- To plan technology refresh and software upgrades
- To plan release and deployment packages and migrate service assets to different locations and service centres
- To optimize asset utilization and costs, e.g. consolidate data centres, reduce variations and re-use assets.

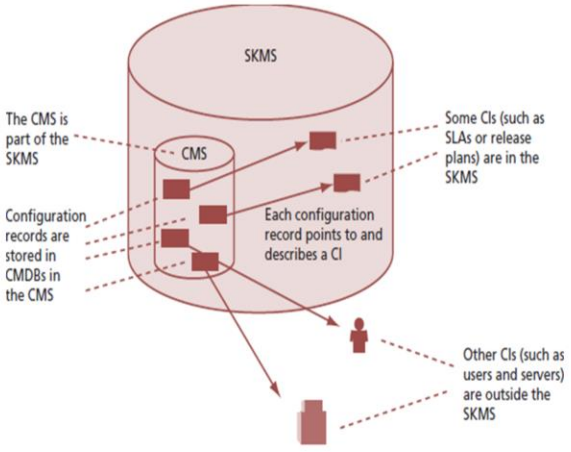
The real power of Configuration Management's logical model of the services and infrastructure is that it is THE model – a single common representation used by all parts of IT Service Management, and beyond, such as HR, finance, supplier and customers.

The configuration items and related configuration information can be at varying levels of detail, e.g. an overview of all the services or a detailed level to view the specification for a service component.

Configuration Management should be applied at a more detailed level where the service provider requires tight control, traceability and tight coupling of configuration information through the service lifecycle.

Configuration Management System (CMS)

- The CMS maintains the relationships between all service components and may also include records for related incidents, problems, known errors, changes and releases
- The CMS may also link to corporate data about employees, suppliers, locations and business units, customers and users



To manage large and complex IT services and infrastructures, service asset and configuration management requires the use of a supporting system known as the configuration management system (CMS).

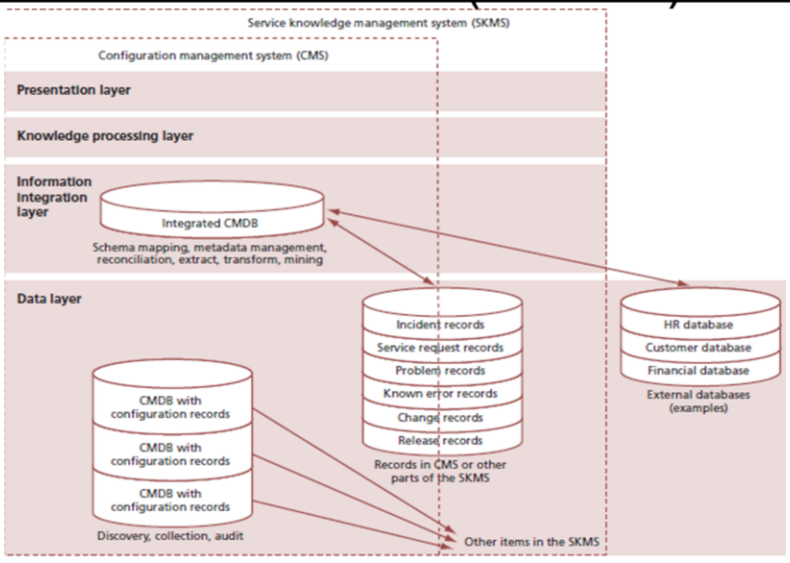
The CMS is also used a for wide range of purposes: for example, asset data held in the CMS may be made available to external fixed asset management systems to perform financial reporting outside service asset and configuration management.

The CMS maintains the relationships between all service components and may also include records for related incidents, problems, known errors, changes and releases. The CMS may also link to corporate data about employees, suppliers, locations and business units, customers and users; alternatively, the CMS may hold copies of this information, depending on the capabilities of the tools in use.

The CMS holds all the information about CIs within the designated scope. Some of these items will have related specifications or files that contain the contents of the item (e.g. software, document or photograph), and these should be stored in the SKMS. For example, a service CI will include the details such as supplier, cost, purchase date and renewal date for licences and maintenance contracts; related documentation such as SLAs and underpinning contracts will be in the SKMS.

The figure above shows the relationship between configuration records, stored in the CMS, and the actual CIs, which may be stored in the SKMS or may be physical assets outside the SKMS.

Configuration Management Databases (CMDBs)



At the data level, the CMS may include data from configuration records stored in several physical CMDBs, which come together at the information integration layer to form an integrated CMDB. The integrated CMDB may also incorporate information from external data sources such as an HR database or financial database. Since this data is normally owned by other business units, agreements (OLA) will be needed about what data is to be made available and how this will be accessed and maintained. Records used to support service management processes, such as incident records, problem records, change records, release records and known error records, must be associated with the specific CI that they relate to. These records can be included within the CMS; or in the SKMS but outside the CMS. What is important is that there are appropriate links to enable all of these records to be located and searched as required to support the delivery of services.

The figure above shows the architectural layers of the CMS. The presentation layer of the CMS will contain views and dashboards that are required by people who need access to configuration information, for example:

- **Change and release view** Used by personnel responsible for change management and release and deployment management
- **Technical configuration view** Used to support the needs of personnel in technical and application management functions
- **Service desk view** For use by the service desk, for example when logging and managing incidents and service requests
- **Configuration lifecycle view** Used by service asset and configuration management personnel who are responsible for managing the lifecycle of configuration items.

The CMS typically contains configuration data and information that combines into an integrated set of views for different stakeholders through the service lifecycle. It therefore needs to be based on appropriate web, reporting and database technologies that provide flexible and powerful visualization and mapping tools, interrogation and reporting facilities. The CMS is part of an SKMS that includes data, information integration, knowledge processing and presentation layers.

Configuration Baseline and Snapshot

- A configuration baseline is the configuration of a service, product or infrastructure that has been formally reviewed and agreed, which thereafter serves as the basis for further activities and can be changed only through formal change procedures
- A snapshot is the current state of a configuration item or an environment, e.g. from a discovery tool.
  - A snapshot is recorded in the CMS as a fixed historical record (footprint)
  - A snapshot is not necessarily formally reviewed and agreed on

A **configuration baseline** is the configuration of a service, product or infrastructure that has been formally reviewed and agreed, which thereafter serves as the basis for further activities and can be changed only through formal change procedures. It captures the structure, contents and details of a configuration and represents a set of configuration items that are related to each other.

Establishing a baseline provides the ability to:

- Mark a milestone in the development of a service, e.g. service design baseline
- Build a service component from a defined set of inputs
- Change or rebuild a specific version at a later date
- Assemble all relevant components in readiness for a change or release
- Provide the basis for a configuration audit and back-out, e.g. after a change.

A **snapsho**t is the current state of a configuration item or an environment, e.g. from a discovery tool. This snapshot is recorded in the CMS and remains as a fixed historical record. Sometimes this is referred to as a *footprint*. A snapshot is not necessarily formally reviewed and agreed on – it is just a documentation of a state, which may contain faults and unauthorized CIs. One example is where a snapshot is established after an installation, perhaps using a discovery tool, and later compared to the original configuration baseline.

The snapshot:

- Enables problem management to analyse evidence about a situation pertaining at the time incidents actually occurred
- Facilitates system restore
- Supports security scanning software.

Secure Libraries and Secure Stores

- A secure library is a collection of software, electronic or document CIs of known type and status
- A secure store is a location that warehouses IT assets. It is identified within SACM
  - Example: secure stores used for desktop deployment

A secure library is a collection of software, electronic or document CIs of known type and status. Access to items in a secure library is restricted. Libraries are used for controlling and releasing components throughout the service lifecycle, e.g. in design, building, testing, deployment and operation.

A secure store is a location that warehouses IT assets. It is identified within SACM – e.g. secure stores used for desktop deployment. Secure stores play an important role in the provision of security and continuity, maintaining reliable access to equipment of known quality.

The contents of secure stores and secure libraries should be recorded in the CMS, and change management approval is needed to move things into or out of them.

Secure Libraries and Secure Stores

- A secure library is a collection of software, electronic or document CIs of known type and status
- A secure store is a location that warehouses IT assets. It is identified within SACM
  - Example: secure stores used for desktop deployment
- Types of secure libraries and stores:
  - Definitive spares
  - Definitive Medial Library (DML)

A secure library is a collection of software, electronic or document CIs of known type and status. Access to items in a secure library is restricted. Libraries are used for controlling and releasing components throughout the service lifecycle, e.g. in design, building, testing, deployment and operation.

A secure store is a location that warehouses IT assets. It is identified within SACM – e.g. secure stores used for desktop deployment. Secure stores play an important role in the provision of security and continuity, maintaining reliable access to equipment of known quality.

The contents of secure stores and secure libraries should be recorded in the CMS, and change management approval is needed to move things into or out of them.

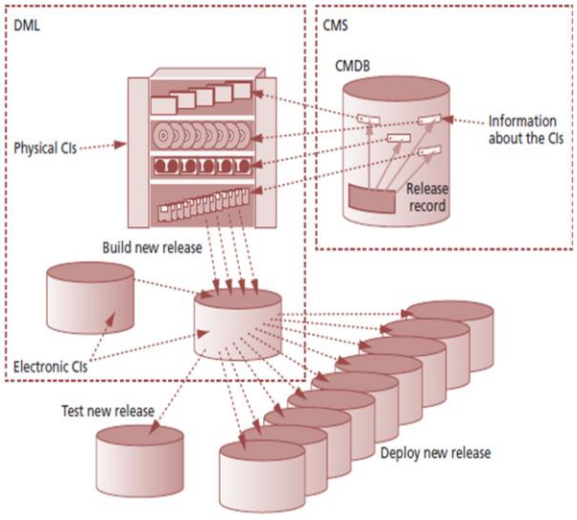
Definitive spares

An area should be set aside for the secure storage of definitive hardware spares. These are spare components and assemblies that are maintained at the same revision level as the systems within the controlled test or live environment. Details of these components, their locations and their respective builds and contents should be comprehensively recorded in the CMS. These can then be used in a controlled manner when needed for additional systems or in the recovery from incidents. Once their (temporary) use has ended, they are returned to the spares store or replacements are obtained.

Definitive spares should be managed in the same way as other fixed assets, including adherence to policy and procedures for procurement, lifecycle management and disposal.

Definitive Media Library (DML)

- DML is the secure library in which the definitive authorized versions of all media CIs are stored and protected.
  - Stores master copies of versions that have passed quality assurance checks.
  - Stores electronic form of master copies of controlled documentation for a system
  - include a physical store to hold master copies, e.g. a fireproof safe.
- Only authorized media should be accepted into the DML, strictly controlled by SACM.
- Electronic assets in the DML are held within the SKMS, and every item in the DML is a CI.



The definitive media library (DML) is the secure library in which the definitive authorized versions of all media CIs are stored and protected. It stores master copies of versions that have passed quality assurance checks. This library may in reality consist of one or more software libraries or file-storage areas, separate from development, test or live file store areas. It contains the master copies of all controlled software in an organization. The DML should include definitive copies of purchased software (along with licence documents or information), as well as software developed on site. Master copies of controlled documentation for a system are also stored in the DML in electronic form. The DML will also include a physical store to hold master copies, e.g. a fireproof safe. Only authorized media should be accepted into the DML, strictly controlled by SACM. The DML is a foundation for release and deployment management. Electronic assets in the DML are held within the SKMS, and every item in the DML is a CI.

The exact configuration of the DML is defined during the planning activities. The definition includes:

- Medium, physical location, hardware and software to be used, if kept online
- Naming conventions for file-store areas and physical media
- Environments supported, e.g. test and live environments
- Security arrangements for submitting changes and issuing documentation and software, plus backup and recovery procedures
- The scope of the DML, e.g. source code, object code from controlled builds
- Archive and retention periods
- Capacity plans for the DML and procedures for monitoring growth in size
- Audit procedures
- Procedures to ensure that the DML is protected from erroneous or unauthorized change
- Procedures to ensure that the DML is backed up