Part Five: IT Project Management

12. Release And Roll-Out Management

Given an IT Software Project, we can consider Release Management to be that management discipline which is closest visible for the customer - and impacts him most important. While preparing this script a 'race' is ongoing about the next generation Web Browser among the combatants Microsoft's IE7, Firefox 3, and finally Opera 9.5 (Apple's Safari does not play a relevant rôle under Windows). This order reflects the market shares, however, the release order is inverted, and reflects probably the pressure from the marketing department (figure 88).

Despite of this Release and Roll-Out Management is not directly covered by any Project Management Framework, however it is defined as ITIL Service Management process. Often, Release Management is considered as part of Change Management, though this does not cover the complexity of the Release Management process entirely.

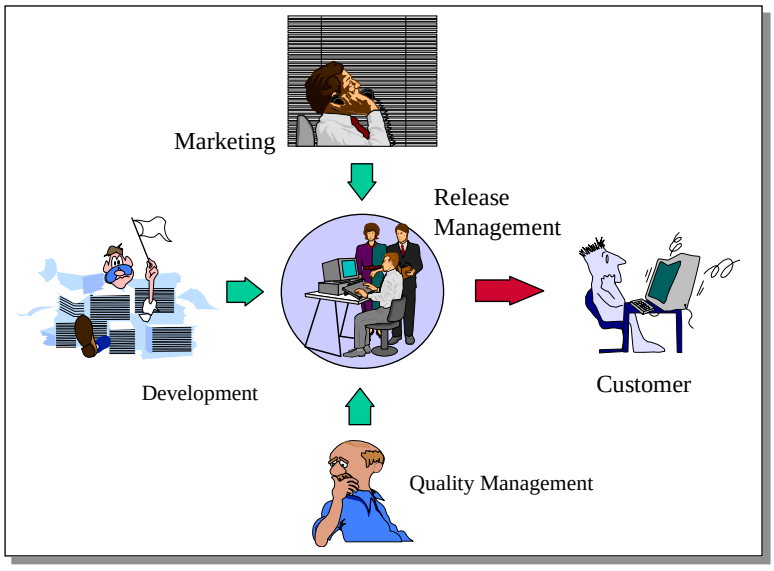

Figure 88: Release Management as key discipline for customer satisfaction

12.1 Release Management a la ITIL

Release Management is of part of ITIL's (IT Infrastructure Library) Service Management Functions SMF as shown in figure 89. According to ITIL's understanding Release Management is required for

- the introduction of substantial and critical hard changes,
- the deployment of relevant software,
- bundling complex and interdependent changes.

The Release Management has the task of co-ordinating the Service Providers, delivering companies, and other involved third parties in order to integrate all products subject for deployment and complement them with the required documentation and perhaps training.

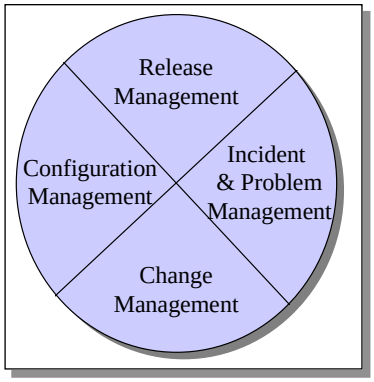

Figure 89: ITIL Service Support Management Functions (SMF)

A central concept of ITIL is the Definitive Software Library DSL. This serves as repository for the code, while configuration parameters are stored in the Configuration Management Database CMDB.

> *Note, that the DSL is not equivalent to the database of our Software Version Control System (VCS) and the CMDB is not the same as our Document Management System (DMS).*

The release process has to be guided by some principals of conduct. In particular, the rôles of the different involved parties (figure 88) have to be clearly defined and the responsibilities of the Release Management have to be expressed (see figure 90).

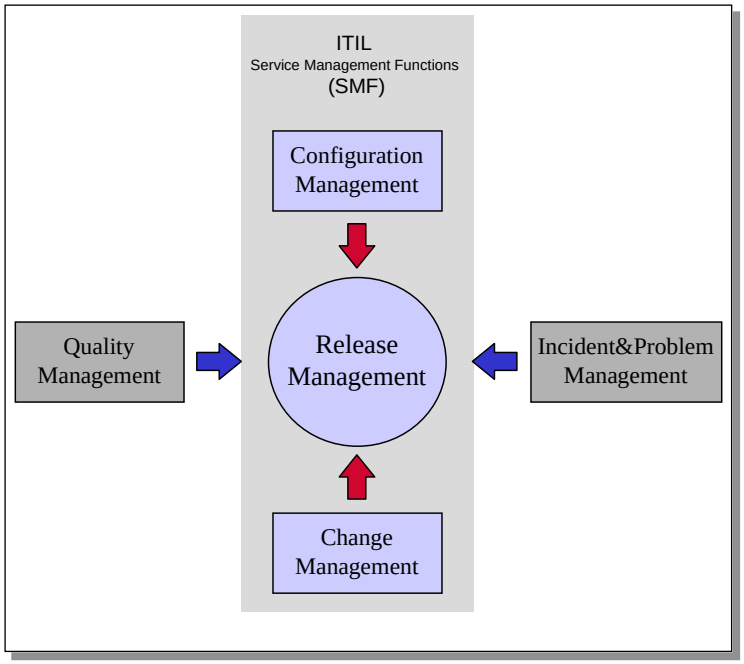

Figure 90: Dependencies of the Release Management with other management disciplines

Regarding ITIL, the main tasks of the Release Management are:

- Planning and supervision of roll-outs considering any changed hard- or software, including all relevant documents.
- Alignment with Change Management regarding the contents of the release and a detailed rôle-out schedule.
- Responsibility that all products subject for release are available and are registered as entries in the CMDB.
- Providing release information for customers in order to prepare the acceptance of the products and to shape the expectations.

Regarding the forthcoming release, ITIL recognises the following release-types:

- *Full Release*: All components have been commonly developed, tested, deployed and implemented.
- *Delta Release:* Only particular changes, encountered since the last release are considered.
- *Package Release:* Is a release superset, including different products (even third party) and full or delta releases.

ITIL does not consider software development and quality management explicitly in it's functional model, however makes the following recommendations:

Build Management

The soft- and/or hardware subject for a release have to 'build' in a consistent manner, allowing a reproducible processes. Some processes have to be automated in order to improve consistency and to minimise human mistakes. As soon, as the test-phase starts, Build Management should be the responsibility of Release Management.

Test Procedures and Fall-Back Strategies

Any release has to be carefully tested, prior of the approval, the user acceptance has to be verified. In addition. Fall-back strategies and plans have to be prepared, documenting what steps are required, in case the release and/or the final deployment is error-prone. The fall-back strategies and plans have to be tested in the reference environment.

12.2 Release Management Overview

Alternatively to ITIL's approach within Project Management we can consider Release Management as part our quality approach and thus belongs to Quality Management (compare figure 88). While Quality Management interfaces with development, Release Management interfaces mainly with the customers.

Figure 91 shows the dependencies between Release Management, Software Development, Quality Management, and Change Management. We identify, that Release Management is an essential part of the *Software Delivery Life Cycle* ***SDLC***.

Change Advisory Board

Typically, the Release Manager is member of the *Change Advisory Board* ***CAB***. More complex releases (i.e. an Operating System) will probably employ different Release Managers for the different products subject to release. Other members of the CAB are the Quality Manager(s), the head of Software Development, and

perhaps representatives (*bridgeheads*) of other companies, in case of third party software/hardware.

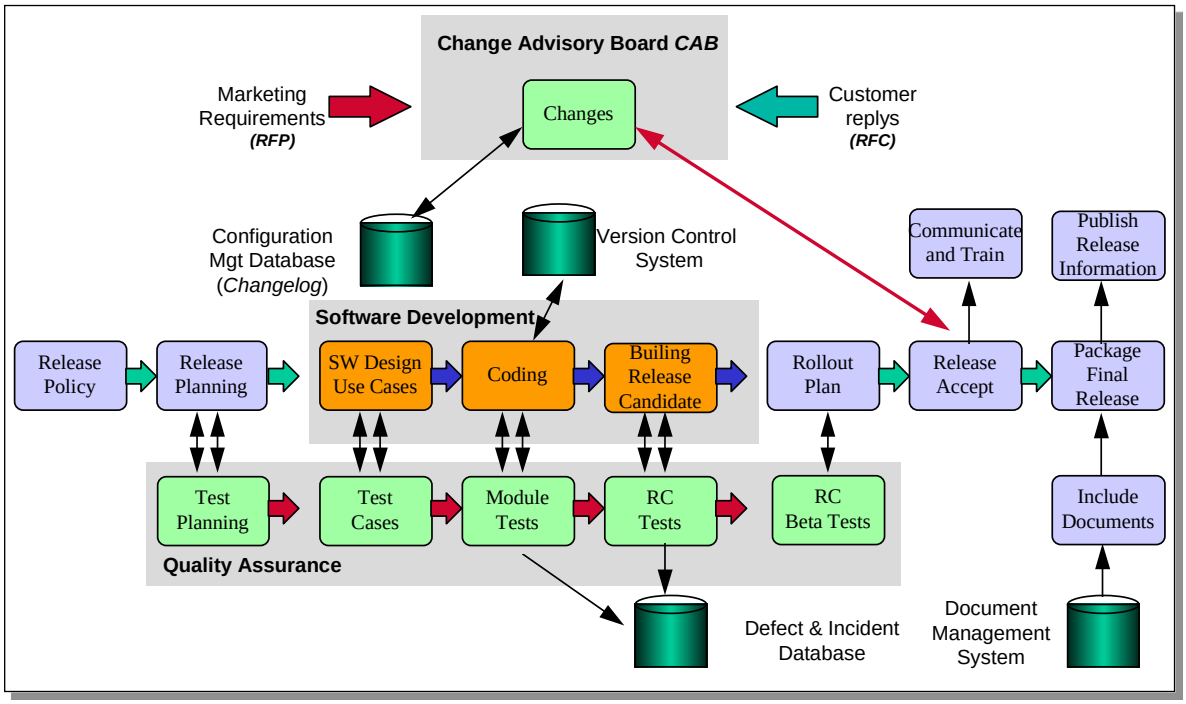

Figure 91: Release Management steps from design to final release

Software Development

While software development can be facilitated according to the models as discussed in the previous chapter, the Release Manager has finally to identity each software component to be included in the final release. Probably. within the software development department, somebody maintains the content of the Version Control System VCS. The last step of the development process is to label in the VCS the individual components and modules. The Release Manager and the head of Software Development have the common responsibility to ensure the completeness and the consistency of the release.

Quality Assurance

The main part of the quality management process belongs to the software development phase or stage. However, the Quality Manager plays an import rôle to guarantee the absence of major bugs in the release. Unfortunately, a 100% bug-free delivery is almost impossible and thus the Quality Manager has in the advent of a forth-coming release already to prepare resources to support post-release bug-fixing.

A special case has to be considered if the software product will be deployed on different systems, ie. levels of the OS or other potentially impacting causes. Here, software development and quality management can decide to use a controlled field test, typically known as "beta tests" in contrast to the in-house "alpha tests".

12.3 Release Planning

Release Planning is different with respect to the initial release and the forthcoming releases. The release initiators and the context for next release in addition with the release request type are summarised in table 8:

Rôle	Context	Release Entry Point	Release request type
Senior Executive	Discussions of largest grained needs. Final association of most serious issues ideation of exporation.	Senior IT Leader	Usually personal interactions and unstructured docuents (emails, presentations, etc.). Request for information (RFI). Starts with personal interactions and moves into process driven.
	Requests for major new systems. Large project level.	IT CRM	RFP - Request for Proposal.
Unit Executive	Requests for major new systems. Large project level.		RFC (high level) Request for Change.
		Demand Management System	Starts as process-driven demand request or RFC
Functional area owner	Requests for new systems, additonal functionality	Demand Management System	Starts as process driven demand request or RFC.
		IT Service Request System	Development requirements.
User	Requests for orderable IT Services Reporting of Incidents	Incident Management System	Process driven for service requests, reporting requests, Incident reports, or Change RFCs (operational)
		Change Management System	Management system. New system requirements.

Table 8: Releases Sources and Initiators [DrapeauOudi2007]

Initial Release

Even before the first modules are coded, Release Management should define how releases are identified and how the releases are realised in the VCS. Thus choice of the VCS and it's capabilities determine to some extend the flexibility of the release process.

- The Identification of a release can be quite difficult and is a little bit arbitrarily. Initially we have the following information:
- The Version of the different software modules; either provided the VCS or by the software author him/herself.
- The Version(s) and perhaps Release(s) of external components which have to bundled with the current release.
- The Build Number of the VCS which labels a particular configuration or counts the steps for compilation/linking, thus to generate the executable modules.

For efficient defect tracking all these information have to be included in the Configuration Management Database CMDB and are now commonly referred to as

"Release". In order to support the SDLC, we typically use the following hierarchical scheme:

- *Major Release*
- *Minor Release*
- *Fixlevel*

and assign a number to a particluar release 2.7.18. As discussed above, a Minor Release could be realised as Delta Release and will only install against a particular Major Release. Occasionally, this is also called a V.R.F. scheme, with V = Version, R = Release, and F = Fixlevel. In this way, we can preserve the logic and the dependencies of the software development.

In addition, it might be useful to include a *Delivery Information* as well. In some cases, the delivery information is appended by a dash "-" or underscore "_" sign. For instance a release could be introduced as 1.2.0_02, telling the customer that this is the second ('02') delivery of release 1.2.0. While the code base shall be the same for 1.2.0 and 1.2.0_02, maybe because some packaging errors occurred, or because extra documentation has been added a first and second delivery is necessary.

Further, some companies tend to include the *Build Number* in the release information (i.e. for the Windows NT OS while preserving a descriptive name for the development). Other companies use additional names to a specific release (i.e. Apple for the OS X - Panther: 10.3, Tiger: 10.4, Leopard 10.5).

Further Releases

The release planning for the forthcoming releases is mainly triggered from four sides:

- *Market*: Marketing has identified a set of requirements and 'trends' to follow.
- *Customers*: The customers report defects and express need for additional functionalities.
- *Quality Management*: The open defects in the previous release needs additional fixes.
- *Technical Progress (Development)*: Changing standards have to be incorporated into the next release.

Here, we have to consider two main processes:

- *RFP - Request For Proposal*
  will typically include an initial definition for additional requirements
- *RFC - Request for Change*
  are issued by the Change Manager and need approval by the C*hange Advisory Board CAB*. All realised changes need to be documented. Prior for the next development round the *Change Advisory Board* has to digest the requested changes for the product and finally decide what to include into the next release. Changes have to be identified by module and impact. Apart from the sizing, they are also prioritised.

Thus, the release plan will include a development timeline, where changes are well defined for the next releases but will become more and move vague.

12.4 Code And Patch Management

Code and Patch Management is a task of Software Development, however during a release build, the Release Manager needs to audit this process. Typically, it is the software developer who modifies the code on it's own behalf. In particular for Open Source projects, code changes will also arrive by external developers or users. In addition, field engineers may change the code on customer demand, in particular to fix a problem.

Changes to the source code or the executable module which modify the execution are called *Patches*. In case the patch is realised against the executable program, it needs to be re-engineered and included into the mainstream.

Code changes and patches which impact the functional behaviour of any module (and thus fix a defect) have to consider four different issues:

- What is the reason for the code change ?
- Who is the maintainer of the code ?
- To which version of the software do we need to apply the change, thus in what release should the enhancement/change be incorporated ?
- Does this change impact Use and/or Test Cases ?

Figure 92 shows the inclusion of patches into the development cycle until final inclusion.

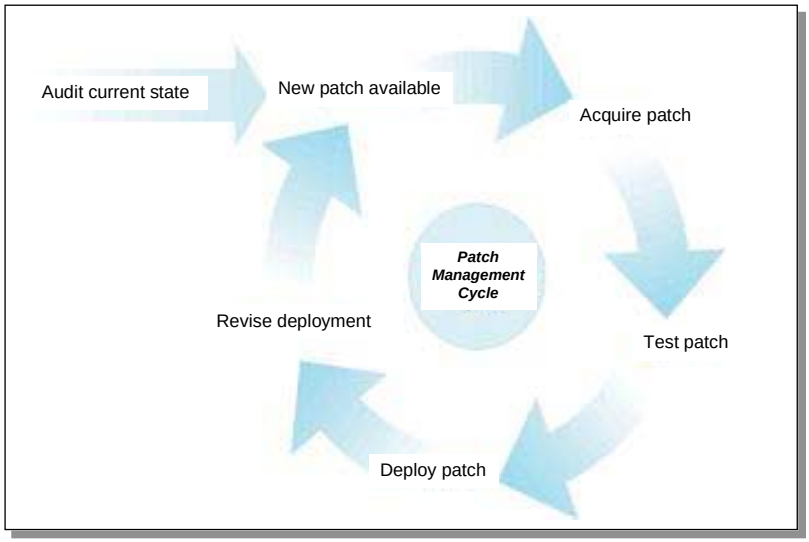

Figure 92: Patch Management Cycle [DrapeauOudi2007]

12.5 The Build Process

The developers will maintain and develop the code according to the releases. In a VCS, the developers can maintain a 'view' of the particular releases. Software modules are checked in and checked out against a certain release. Thus we have to consider two distinct logical views:

- The *Branch* - this is a logical order software modes in the software repository, the VCS, common for all users.

- The *View* - this is the workspace of the developer; a view may consist of different branches.

It is the requirement of the Release Manager to determine when the current branch (see figure 93) should be forked, thus a new (development) release is due. This decision impact the consistency of the code base:

- In case of a too early branch, often changes in the old branch have to applied to the new branch.
- In case of a too late branch, development capacities are still focused on the old release, thus the new release may be delayed.

Branching

Release Management has to realise, that each release (=branch) has its own SDLC, thus generally, any branching should be avoided. Of course, this binds resources and causes efforts, perhaps not planned initially.

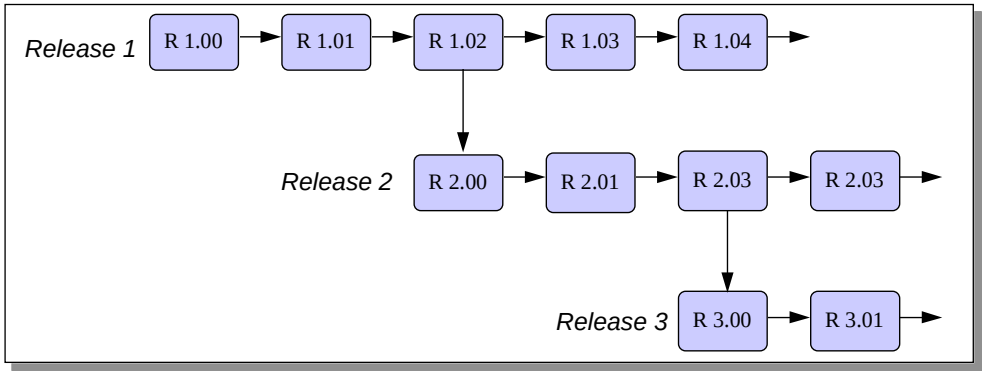

Figure 93: Branching into different releases in the VCS [~ Neuma]

As a consequence, the developer may occasionally modify modules in a wrong branch, since this information is typical not visible for the developers, since they are assigned to a particular release/branch. It is the obligation of the Release Manager or head of software development, first to communicate with the developers regarding the release schedules and second to correct wrongly checked-in modules (figure 94).

Labeling

All software components which should be included in the release are labeled. Effectively, even modules can be labeled for a certain release, which belong to a different branch.

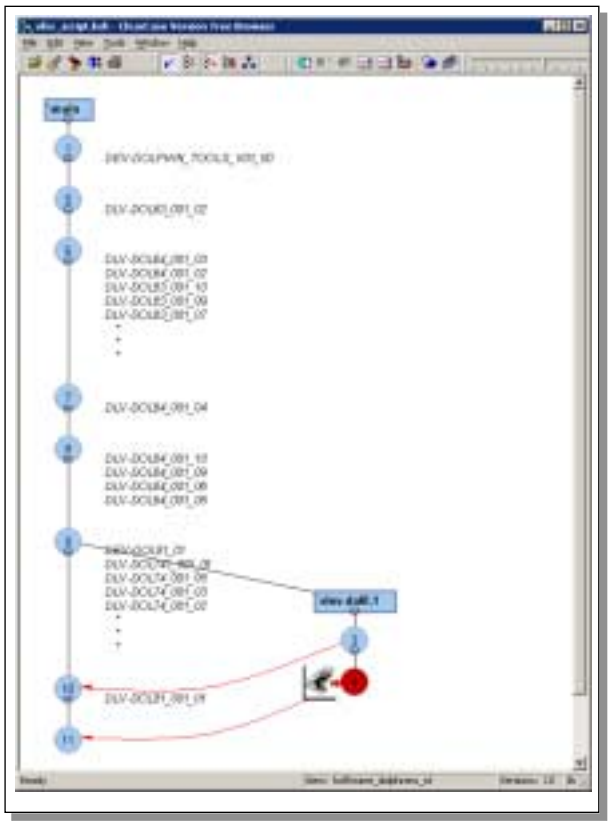

Figure 94: Including a changed module into a specific branch

12.6 Quality Assurance

The strong dependencies between Release and Quality Management can be depict from figure 95.

Figure 95: Quality Assurance for the Release process [DrapeauOudi2007]

As discussed, both the Quality Manager and the Release Manager have to be prepared about the remaining defects in the current release. Figure 96 shows on a time

line the 'creation' and fixing of defects. By means of a good quality management system, most defects can be corrected before the actual product comes into operation.

- Most probably, fixes for the remaining defects are scheduled for the next (minor) release. However, it might be necessary to deploy important bug fixes in a particular Hotfix release.

There exist different opinions about Hotfixes. Occasionally, they are part of the standard release cycle or they are real "hot-fixes" required for special customer problems/issues. Important is the recognition of a hot-fix for further enhancements. Typically, hot-fixes are 'stand-alone', where as in particular any minor release will recognise the existing installation base and update this respectively.

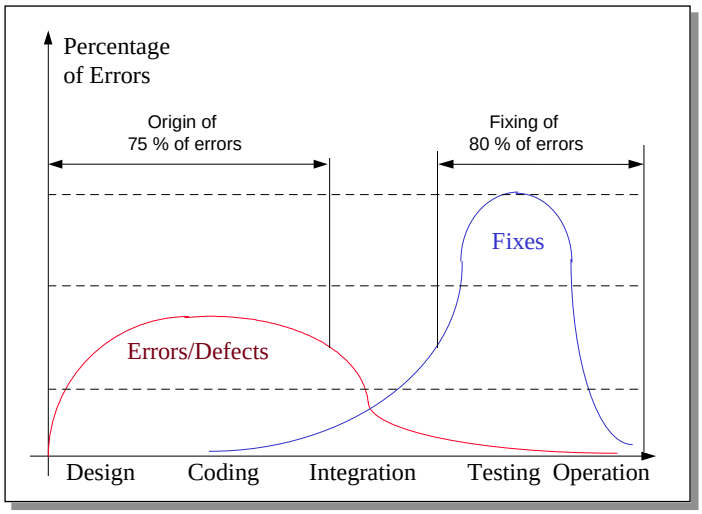

Figure 96: Defect appearance and fixing during the development and test cycle

12.7 Release Readiness

Declaring Release Readiness is responsibility of the Release Manager. However, the final approval is done by the *Change Advisory Board*. Depending on the current state of completion and defects, the CAB may conclude to issue a *Release Candidate* (***RC***) prior of the final release (figure 97).

The Candidate build may be issued to:

- the QA department
- the public or interested customers/users.

While the QA can determine the readiness according to a well-defined environment, a 'public RC' does not; however it can be used to gather experience and perhaps to identify design problems, not recognised yet.

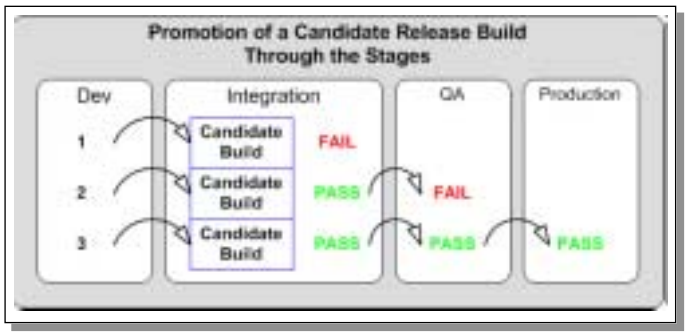

Figure 97: Building Release Candidates

12.8 The Roll-Out Process

The Roll-Out process may include several software and even hardware products. For example, today's OS are couples with email-systems, web-browsers, text-editors, multi-media-software and other 'gadgets'. Thus within the roll-out process different products need to be bundles as part the Release Management.

Additionally, the Roll-Out process has to include the required release documents. Depending on the product and the targets may include:

- Installation documentation
- Release documentation
- User manuals
- Administrator guides

Once the roll-out is prepared, the medium for the roll-out has to be set-up. While 'in the old days' software releases are shipped on diskettes or CDs, nowadays, everyone accepts downloads from a particular Download Server. In any case, it has to be verified by the Release Manager, hat the required resources are available

References

[Mircosoft2004] Service Management Functions

[Neuma] Taking the Complexity Out of Release Management

[DrapeauOudi2007] Release Management: Where to Start?