

UNIT - V

Software Release

Compiled by Mr. Gouthaman.P,
Dept of IT, SRMIST, KTR

Disclaimer:

The lecture notes have been prepared by referring to many books and notes prepared by the teachers. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.

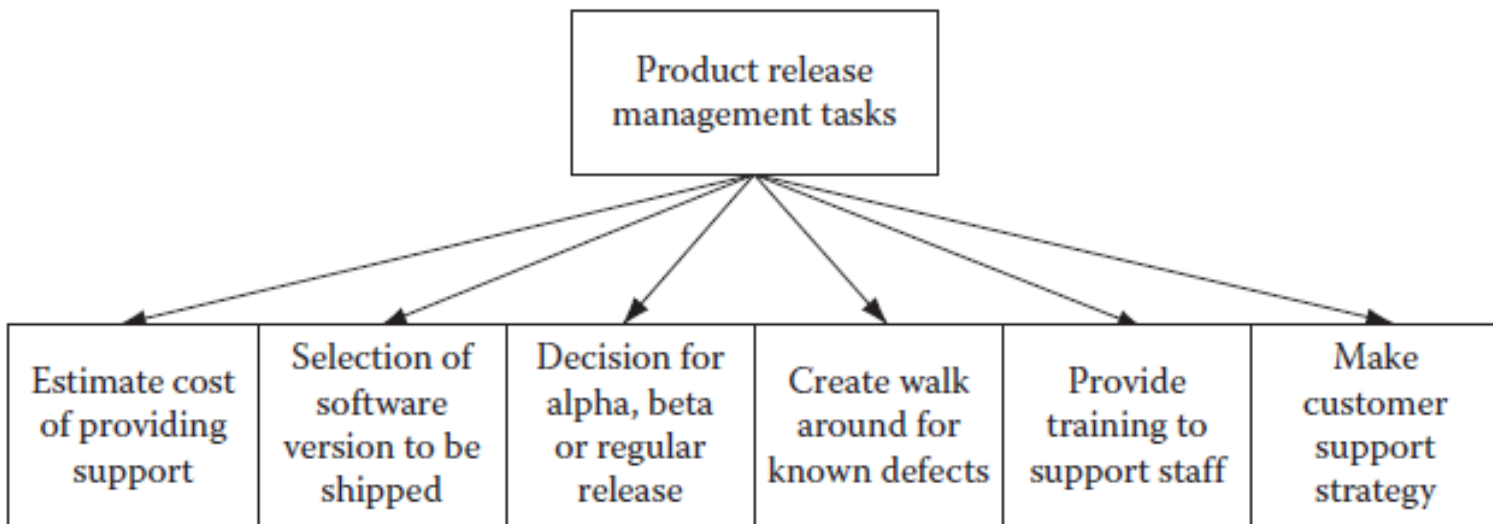
Topics

- Product Release
- Product Release Management
- Implementation
- User Training
- Introduction to Maintenance
- Maintenance Types – Corrective, Adaptive, Perfective, Preventive
- Maintenance Cost
- Maintenance Process
- Life Cycle
- Software Release and Software Maintenance

Product Release

Introduction

- The software product which has been built till now is now complete.
- It needs to be taken to the customer's site and get it implemented so that the end users can start using it.
- However, do not run fast in anticipation of wrapping things as early as possible.
- After all this is the magnum opus and needs to be carefully handled.
- It is necessary to make sure that all the tasks are completed, for example, product support cost estimate, walk arounds for known bugs, which version of the software product to be released, if release should be an alpha, beta, or normal release, training needs fulfilled, and customer support strategy (Below Figure-Task list for software product release).

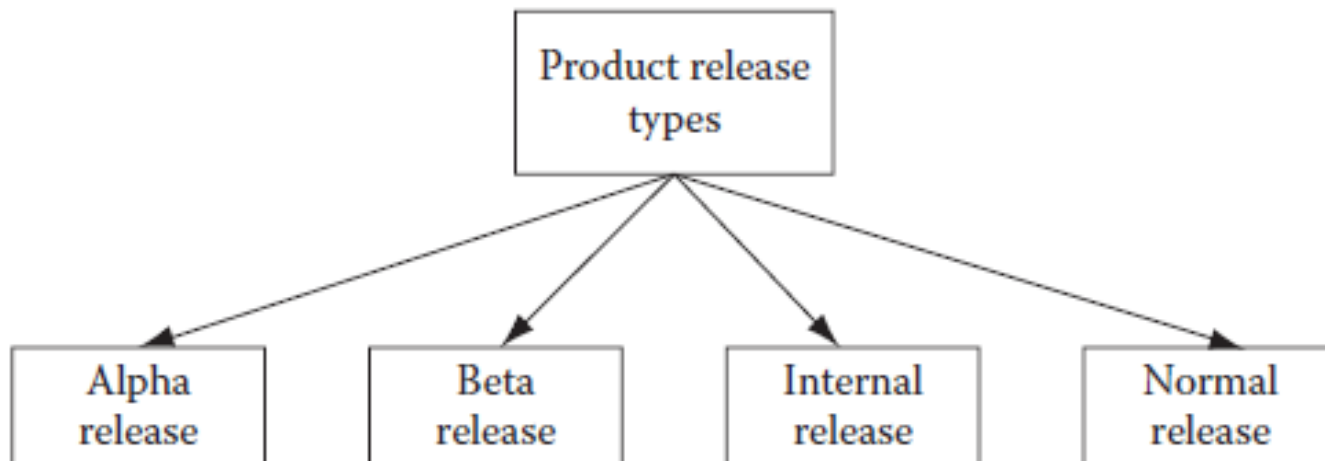


Product Release Management

- Project teams working for software product vendors struggle to keep pace with release of the software product.
- There is pressure from the market to launch new versions by certain dates.
- New features are to be added, porting the product to new platforms, old features are to be enhanced, existing bugs are to be removed, and yet it has to meet the deadline.
- It is a constant struggle that calls for good product release strategies.
- Depending on the situation, the project manager may need to convince the management to cut short some of the product features to meet the deadline as well as meet quality standards.
- A half-baked product will never have any takers; instead the project manager may be blamed for its poor quality issues.
- Bargaining also has to be done for other requirements of bug fixes, feature enhancements, etc.
- If quality concerns are paramount, then moving some of the tasks of new features to a future release may be the best solution for meeting quality standards.

Product Release Management

- If the software vendor is not too sure about product quality, then he may opt for an alpha or beta release of the product.
- In that case, the product will be released only among a few selected groups and not in the market as a whole.
- The controlled product release is the best option in these conditions (Figure below – Software product release types).
- In fact, product release management is such a dynamic environment that if proper planning is not done at a minute level and constant vigilance is not applied over project activities, then a huge mess can be created and there will be no time to clear it.
- So the project manager must be vigilant all the time (Figure below-Software product release types).

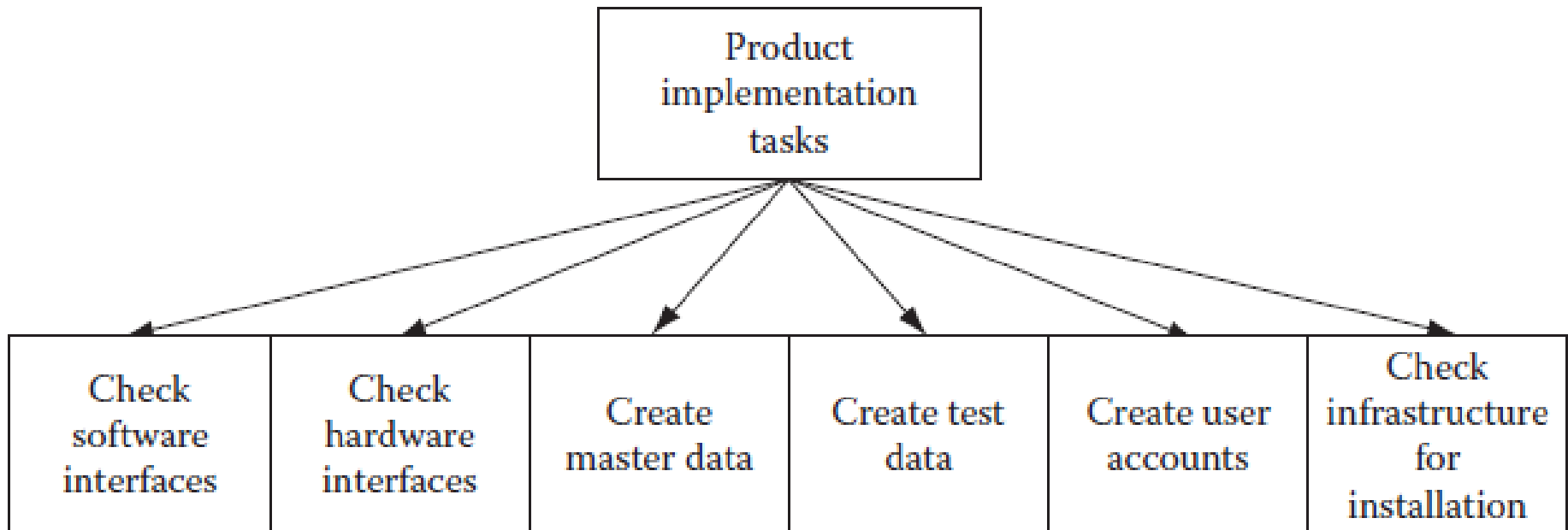


Product Release Management

- Finally, for the product's scheduled release, how the customer support will be provided should be chalked out.
- Walk arounds for known issues, estimated number of critical bugs still remaining in the product, training for the support staff, etc., should be done.
- The cost of support, depending on the number of estimated users, walk arounds, and remaining bugs should be figured out.
- These measures will ensure that the product is transitioned into market without facing major difficulties.

Product Implementation

- The product that has been developed and thoroughly tested now it needs to be implemented at a customer site.
- It is necessary to prepare all master data and test transaction data for testing the implemented product.
- In addition, need to get all required hardware and software that needs to be there for installing the software product.
- Furthermore, need to make sure that development and testing is done for all the hardware and software interfaces for integrating the product, with existing legacy systems and infrastructure.



Product Implementation

- Moreover, need to make sure that the product will run smoothly on customer premises without any interference with their existing applications (Figure above – Task list for software product implementation).
- Often project teams run into problems during implementation, due to unforeseen circumstances or negligence on part of the production team or customer's team.
- Therefore, it is recommended to prepare a list of developer requirements and hand it over to customer's support team so that they are prepared when developers arrive for implementation.

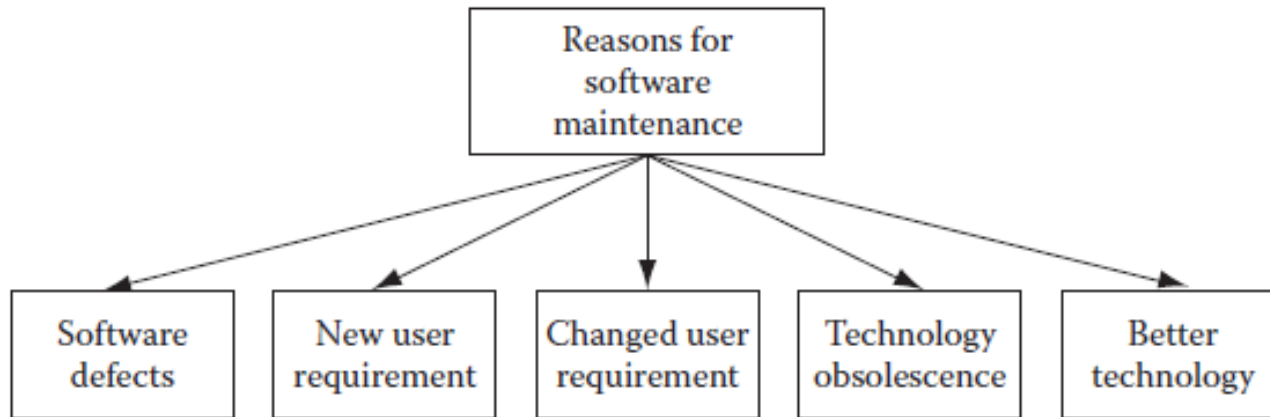
User Training

- Make sure that the user manual prepared by the team is up to date and in synch with the version of the software product, which will be implemented at the customer site.
- It is not possible to provide training to all users.
- So, it is better to prepare a list of roles that are needed to operate the product.
- This list needs to be given to the end users and ask them to select one user per role who will receive the training.
- Apart from the user manual, it is necessary to prepare a tutorial to include probable scenarios that may arise during operation of the product.
- The tutorial will provide a step-by-step guide for using the product under those scenarios.
- This will be a very important step in training, because if users do not learn it during training, then they may contact later after implementation and ask for information as to how to use the product in those circumstances.
- This will lead to a waste of the support team's time.
- It is lot better to train them now, during user training, rather than face user requests later.

Maintenance Introduction

- Software products do not age or wear out like physical products.
- Then why is there a need to have maintenance of software products? Well, there are some factors which make it absolutely necessary.
- Here are some of the reasons:
 1. *Technology obsolescence*: The software platform (operating system, medium of user, interface) or the hardware platform on which the software product runs gets obsolete.
 2. *Software defects*: There are major software defects in the product and it is difficult to operate. For this reason, a software patch may be needed to be applied so that these defects are removed.
 3. *Change in user requirements*: The business organization that was using the software product has seen a change in business transactions or business workflows that are not supported by the software product (Figure below – Reasons for software maintenance).
- It is estimated that more than 70% of all costs associated with software product development, implementation, and support and maintenance is consumed in the activities of supporting and maintaining software products.

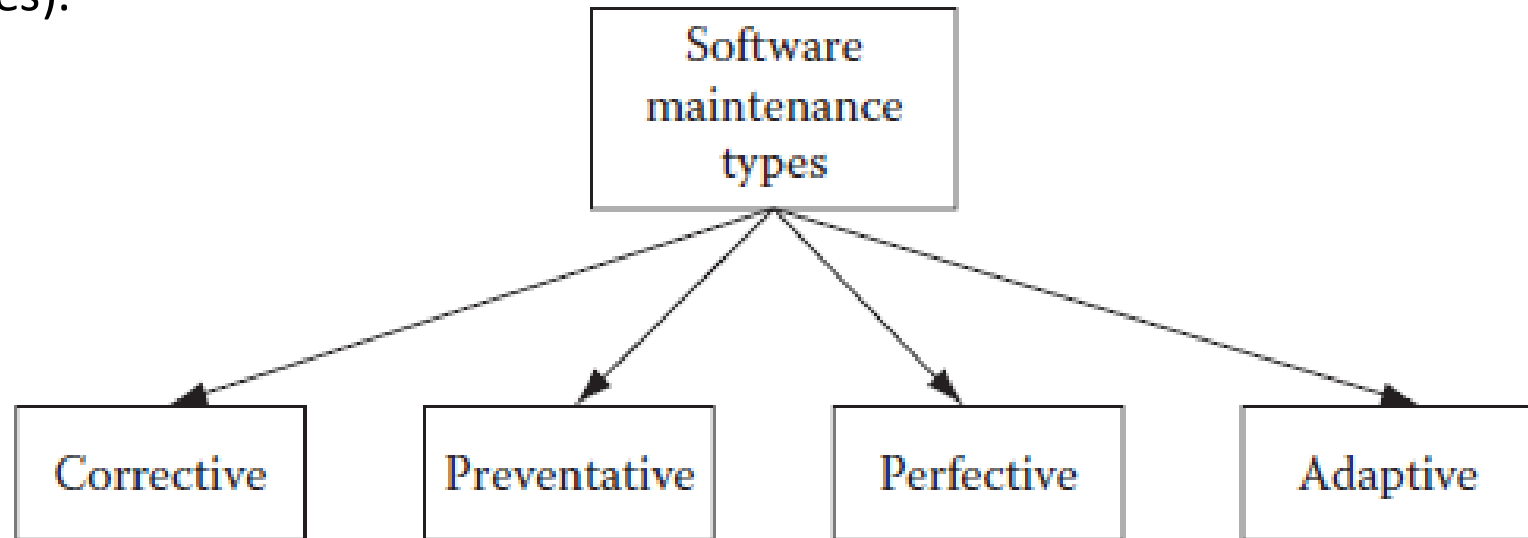
Maintenance Introduction



- Why is it so? What can be done to change this situation, so that support and maintenance costs get minimized as compared to development and implementation costs?
- These kinds of queries have always puzzled the business community.
- This recognition has resulted in an awareness of the importance of finding ways to build such a software product.
- This situation has led to including maintainability characteristics during the entire product development cycle.
- Yet, a lot of work remains to be done during the maintenance phase of any software product.
- How to manage these activities so that costs can be minimized is an area of concern yet to be resolved.

Maintenance Types

- Software maintenance is of four types: corrective, adaptive, preventive, and perfective maintenance.
- If the software has some defects, then it will take a corrective maintenance to rectify it.
- If there are some changes in the operating environment of the software product, then the product can be made useful by doing adaptive maintenance.
- If there is an insecurity that although the product is running fine in future we may have difficulty in using it, then preventive maintenance is employed.
- If there are some deficiencies in the product that must be rectified, then perfective maintenance will fit the bill (Figure below – Software maintenance types).



Maintenance Types

Corrective

- Even after thorough reviews and testing, the software product contains many defects when it goes into production.
- These defects are uncovered as users start using the application.
- They are logged with the support staff and after a sizable number of errors are detected, the software vendor instructs his maintenance team to create a patch to rectify them.
- The maintenance team then makes a plan and fixes those defects.
- After application of the patch containing the fixes, the software starts running without these defects.

Adaptive

- The operating environment in which a software product runs in operation includes the hardware and software platform as well as the interfaces for human and other machine interactions.
- If any of these change over time, it becomes difficult to run the software product.

Maintenance Types

Adaptive

- In such cases, it becomes necessary to do adaptive maintenance so that the software product becomes reusable.
- This kind of maintenance may involve changing the interface or porting the application to another hardware/ software platform.

Perfective

- This kind of maintenance is needed when there is a change in the business environment, and thereby users need additional/modified functionality in the software product to do their tasks.
- A business workflow may have changed, a business transaction may have changed, or an altogether new business transaction was represented in the software product.
- For all these kinds of requirements, a perfective maintenance may be needed.

Maintenance Types

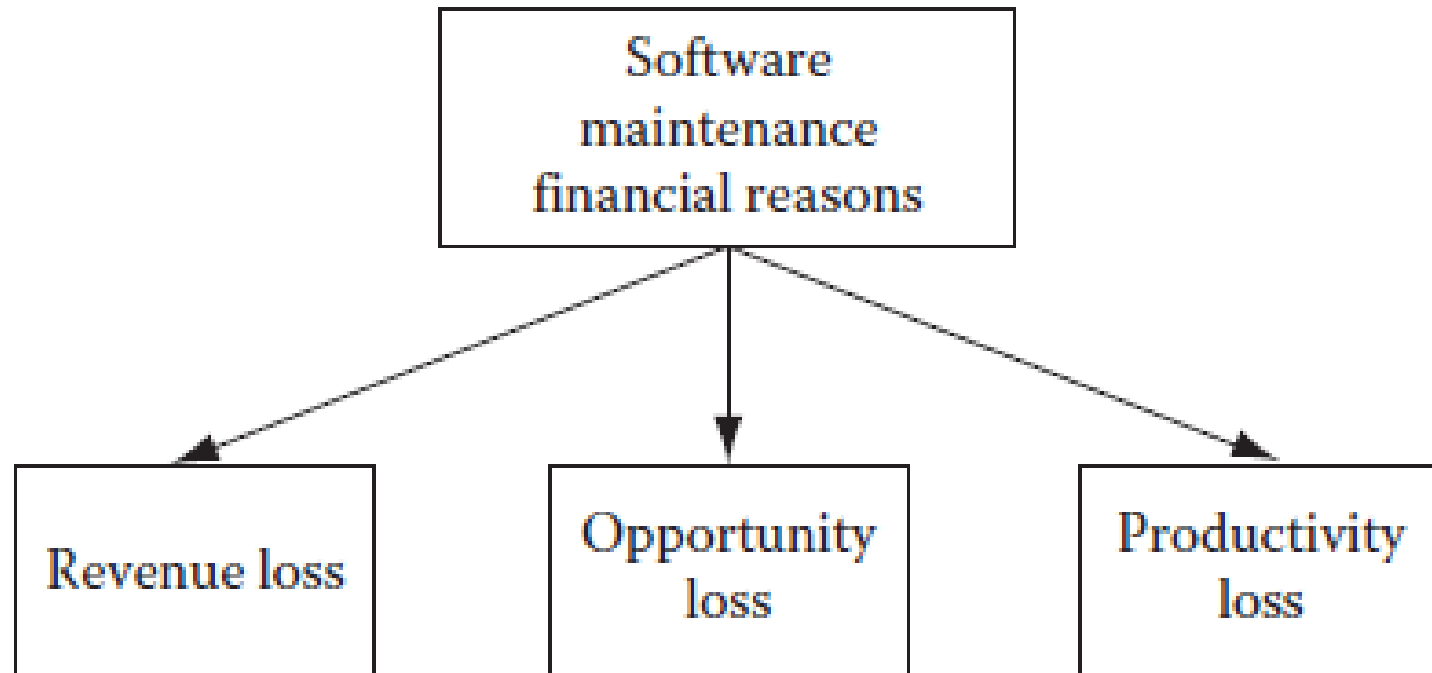
Preventive

- Generally after a lapse of time, there are likely changes in business or operative environment, or there may be changes in hardware/software environment.
- These changes are bound to occur and they affect the way the software product operates.
- Many of these changes can be perceived in advance.
- In such cases, preventive maintenance on the software product can make sure that the product will be useful even after these environmental changes occur.

Maintenance Cost

- A software product is generally very valuable to an organization if it is used for doing a large portion of their daily business.
- If for some reason the software product has become unusable, then the organization in fact will be making losses on their revenue.
- Moreover, large enterprise software products are that much crucial!
- When the organization faces such a case, it is left with no alternative but to either get an entirely different software product that will replace the existing one or do maintenance of an existing product to make it usable.
- Following are some financial reasons for which a maintenance may be needed:
 1. *Loss in business revenue*: It may happen that business transactions are faulty and thus the business may lose revenue.
 2. *Opportunity loss*: Sometimes there could be some business opportunity in the marketplace, but due to some software problems it could not be availed.
 3. *Productivity loss*: If the software product becomes difficult to operate due to many workarounds or lengthy processing then productivity will become lower for business personnel (Figure below – Financial reasons for software maintenance).

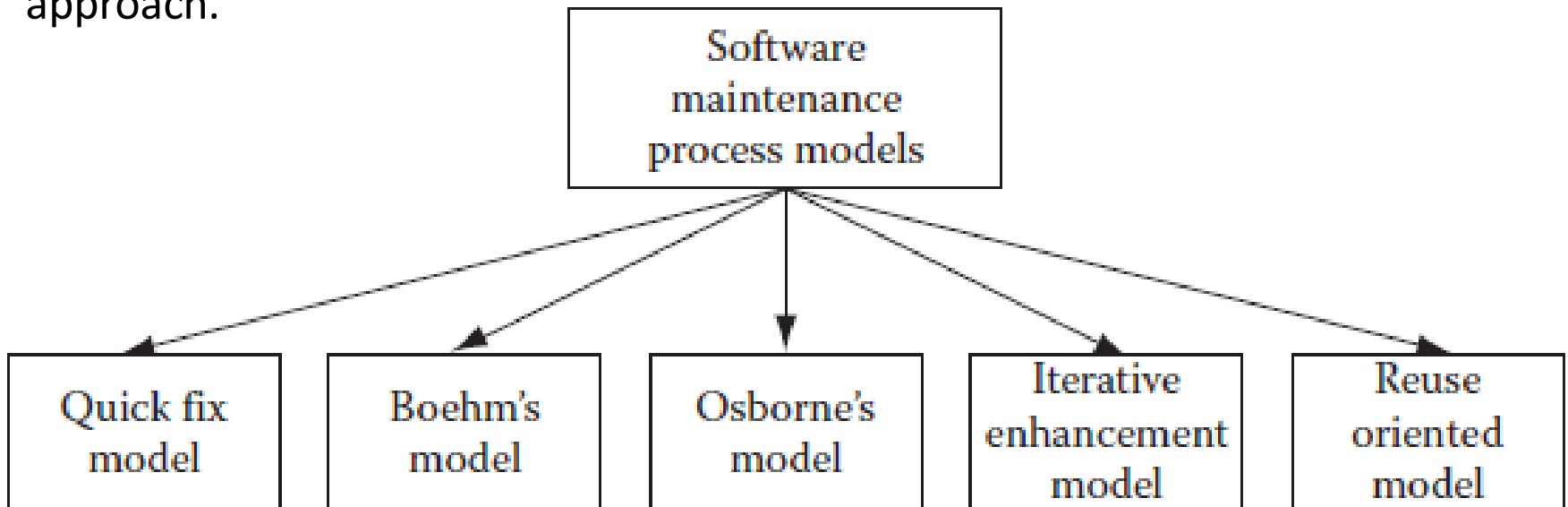
Maintenance Cost



- Maintenance of an existing software product has its own share of problems.
- The maintenance will incur costs.
- A profit/loss analysis can be done, to see if it is more profitable to conduct a maintenance program on the software or keep using it as it is.
- The losses due to problems with the software can be compared to probable cost of maintenance and an ROI (return on investment) can be done.
- If we get a desirable ROI then it is better to go for maintenance.

Maintenance Process

- For any work, it is always better to have a process model instead of doing things on an ad hoc basis.
- When it comes to software maintenance, some process models have been defined.
- Some of the popular ones include the quick fix model, Boehm's model, Osborne's model, iterative enhancement model, and reuse oriented model (Below figure - Software maintenance models).
- *Quick fix model*: This is the simplest of maintenance models; whenever any defects with the software products are found they are immediately fixed. There is no planning involved in the whole process and it is mostly an ad-hoc approach.

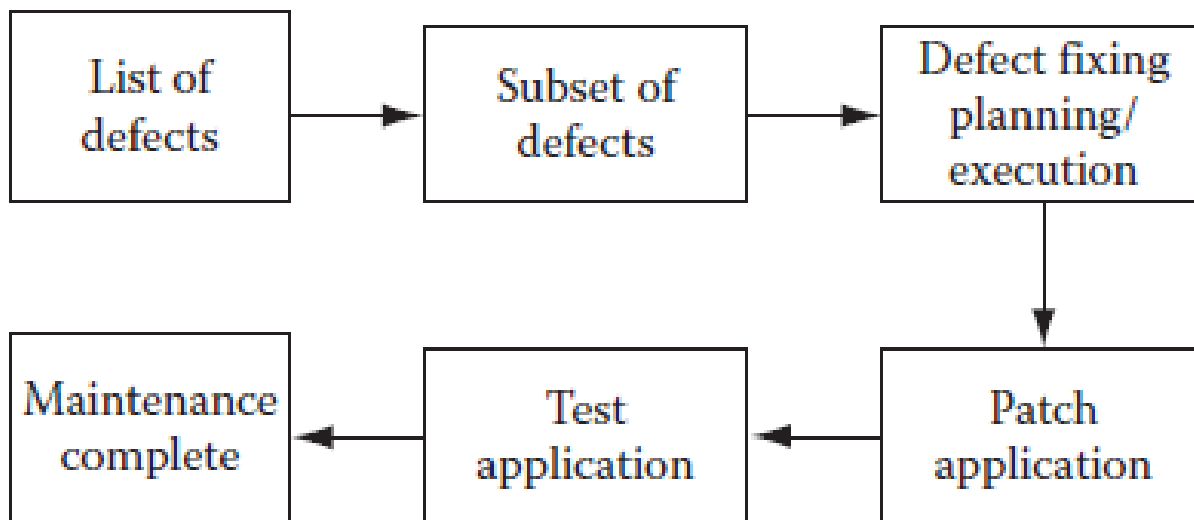


Maintenance Process

- *Boehm's model*: Boehm's model is based on economic models and often involves calculating ROI, for any planned maintenance. If ROI turns out to be good, then it is carried out or else it is dropped.
- *Osborne's model*: Osborne realized that difficulties in carrying out maintenance work are due to gaps in communication. He proposed four steps to prevent this situation. He stated that a maintenance plan should include all change requests in the form of maintenance requirements. A quality assurance plan should accompany the maintenance plan. Metrics should be developed to measure and assess quality of work carried out during maintenance. Finally, reviews should be held after maintenance work to assess quality of work done.
- *Iterative enhancement model*: This model is based on the similar concept of iterative software development. All software defects and change requests are logged and then a small set from this list is taken for making fixes. This set is prepared based on the priority of changes required. High priority fixes are done before low priority fixes.
- *Reuse oriented model*: This type of process is adopted for component-based software products. For fixing any defects, existing components are analyzed and then the appropriate changes are made.

Maintenance Life Cycle

- Like the software development, software maintenance also has a life cycle.
- Requirements for software maintenance come from the list of defects that have been logged.
- Either the list of defects can be taken as a whole or a subset of defects from this list can be taken for a fixing plan.
- It makes a lot of sense to go for an iterative approach.
- This approach is similar to the concept of iterative software development.
- This way it can be ensured that highly visible, important, and priority defects are fixed first and other defects which do not make much impact on operations of the product are tackled later (Figure below - Maintenance life cycle).

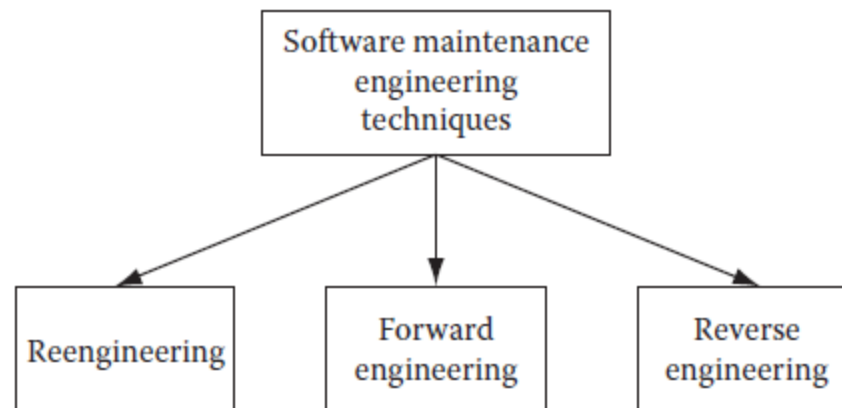


Maintenance Life Cycle

- In the software maintenance life cycle, testing is a crucial phase.
- This phase also consumes a lot of time and effort.
- But the value addition in all this effort and time spent helps in reducing defects, which in the long run is a much cheaper alternative compared to no testing/cursory testing and later spending money in providing support.

Maintenance Techniques

- Maintenance of software products sometimes becomes a tough proposition.
- There is no proper documentation that can be used for understanding how the product is designed and constructed.
- Sometimes there is no documentation at all.
- Even if documentation is there, it is not up to date.
- This out-of-date documentation is not of much use for any maintenance work.
- Sometimes even if the documentation is up to date, the maintenance work is difficult due to dirty design or construction work.
- All these situations call for some specific techniques for maintenance work depending on the situation.
- Some of the common maintenance techniques include reengineering, reverse engineering, and forward engineering.



Maintenance Techniques

Reengineering

- Reengineering is also known as reuse engineering. This technique is a standard method for maintenance work for component-based software products.
- Details about all components in the software products are well known.
- When any maintenance work is needed, from the list of defects, each defect is specifically analyzed to find out the root cause of the defect.
- Once this analysis is successful, then fixing that defect becomes easy.

Reverse Engineering

- Reverse engineering technique is the most useful when nonexistent or sketchy documentation is available for the software product.
- Due to unavailability of documentation, there is no information as to what the design is and how the product is constructed.
- In such a situation, it is almost impossible to do any modification in the source code for any maintenance work.
- In such cases, the reverse engineering technique is adopted.
- Using this technique, similar components or product parts are constructed as compared to existing product components/parts.
- This way the software product functionality is changed as the new constructed parts will have the desired functionality.

Maintenance Techniques

Forward Engineering

- Forward engineering is just the opposite of reverse engineering.
- In this situation, we have ample documentation about the existing product.
- Due to new customer needs, the existing product needs to be extended so that the new needs can be fulfilled.
- All new extended development is based on the existing design and construction methods and will be made for the same hardware/software platform.

Software Release – Case Study

- In the series of case studies, here is the piece related to software release and maintenance.
- SaaS vendor releases minor versions of its product on a quarterly basis and major versions on a yearly basis.
- For each minor release, new features to be added are carefully planned.
- The product manager makes sure that the release plan for a minor release will be on time by assigning priority to each new feature.
- The high priority features will be definitely added and the low priority features for that iteration will be added if any time remains in the iteration.
- SaaS vendor does not release alpha or beta releases of its product as they do not serve mass markets.
- Their product is an enterprise computing product and is used by large retailers, government offices, logistics providers, manufacturers, and distributors.
- They always release new versions of their product to their existing customers.
- Since they do not do alpha or beta releases, they make sure that their new version is tested thoroughly by their testing team, and no major defects are passed in the production instances.
- Since there are no immediate customers who will be available for doing user acceptance testing, the internal testing team does the user acceptance testing as well.

Software Maintenance – Case Study

- The software vendor keeps all of the production instances of its software product at its data centre (also known as operations center).
- All previous versions of the software as well as the current working version of the software product run at this centre as production instances of different versions of their software product.
- The maintenance team makes sure that all versions of the product are available for users.
- They run sanity test scripts daily on all instances.
- If any problems are found, then it is immediately resolved.
- These scripts are run at night.
- If any problems are found then, it is made sure that they are rectified before office hours start and people start using the application.
- In packaged software or custom built software that are not used in an SaaS environment, this kind of quick fix is not possible.
- So in those cases, a maintenance plan is made to fix all or most defects found by users during a time span of 3 months or more.
- But with SaaS environments, this kind of maintenance is not needed at all.
- All defects are quickly and easily fixed, without hampering work of end users.

REFERENCES

- Ashfaque Ahmed, Software Project Management: A Process-driven approach, Boca Raton, Fla: CRC Press, 2012.

THANK YOU