Software Maintenance

by
Manik Chand Patnaik

Software maintenance

- Software maintenance is an important activity for many organizations.
- It is about modifications to a software product after it has been delivered to the customer.

Where Maintenance is Needed?

- Maintenance is inevitable for almost any kind of product.
- Most products need maintenance: due to wear and tear caused by use. But Software products do not need maintenance on this count.
- Software products are maintained to make them fit for new kind of usage and to enhance their usability.

Which software is Maintained?

- Many people think only bad software products need maintenance.
- The opposite is true:
  - bad products are thrown away,
  - good products are maintained and used for a long time.
- There will always be a lot of old software needing maintenance.

Kinds of Maintenance

- There are 3 kinds of software Maintenance :-
  - corrective
  - adaptive
  - perfective

Corrective Maintenance

- Corrective maintenance of a software product is done :-
  - to correct bugs observed while the system is in use.
  - to enhance performance of the product.

Adaptive Maintenance

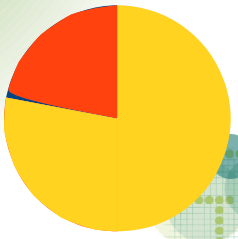
- When customers require a software product to work in
  - A new hardware platform
  - A different Operating System
  - Collaboration with an other software
  - Or in a new context
- Than it needs to adopt to the new environment.
- There is need for the product to interface with new hardware or software or both.

Perfective Maintenance

- Perfective maintenance is required :-
  - to support new features required by users.
  - to change some functionality of the system due suit to customer demands.
  - Ultimately to make the system perfect for use.

Maintenance Effort distribution

- **Corrective 20%**
- **Adaptive 20%**
- **Perfective 50%**



Software Maintenance and Software Evolution

- Every software product continues to evolve
  - through maintenance efforts.
- Larger software products stay in operation for longer time
  - because of high replacement costs.

Laws of Maintenance >

- Lehman's first Law
  - "Software products must change continuously, or become progressively less useful."
- Lehman's Second Law
  - "When software is maintained, its structure degrades unless active efforts are made to avoid this phenomenon."

Click to add title

- Lehman's Third Law
  - "Over a program's life time, its rate of development is approximately constant."
- Other Maintenance Laws :-
  - All large programs will undergo significant changes during operation phase of their life cycle, regardless of apriori intentions.

Legacy Code and Maintenance >

- It is old code, may be Unstructured code
- Maintenance programmers have:
  - insufficient knowledge of the system or the application domain.
  - Because the new Maintenance team is usually different from the development team.

Click to add title

- Documentation may be absent / out of date / insufficient.
  - even after reading all documents it is very difficult to understand why a thing was done in a certain way.
  - there is a limit to the rate at which a person can study documentation and extract relevant information

What makes maintenance difficult?

- Use of goto
- Lengthy procedures
- Poor and inconsistent naming
- Poor module structure
- Weak cohesion and high coupling
- Deeply nested conditional statements
- Functions having side effects (remember global modifications?)

Reverse Engineering & Re-engineering

by

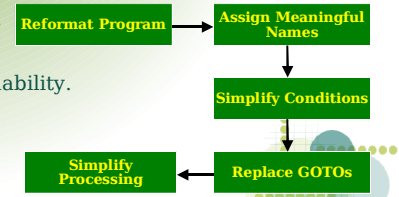
Manik Chand Patnaik

Reverse Engineering

- Reverse engineering is an important maintenance technique :-
- To recover the design and the requirements specification by analyzing a program code.
- It is required because :-
- several existing software products are unstructured, lack proper documentation, were not developed using software engineering principles.

Step - 1

- First carry out cosmetic changes to the code to improve:
  - readability,
  - structure,
  - understandability.



Details of Cosmetic Changes

- Reformat the program:
  - use any pretty printer program
  - layout the program neatly.
- Give more meaningful names to variables, data structures, and functions.
- Replace complex and nested conditional expressions:
  - simpler conditional statements
  - whenever appropriate use case statements.

Other steps

- In order to extract the design fully understanding the code is required.
  - Automatic tools can be used to help derive data flow and control flow diagrams from the code.
- Structure chart is extracted by understanding module invocation sequence and data interchange among modules.
- Requirements specification is extracted by understanding what the code does.

Software Re-engineering

- Re-engineering is a reverse engineering cycle followed by a forward engineering cycle with as much reuse as possible from existing code and other documents.

When applicable?

- Preferable when:
  - amount of rework is significant
  - software has poor structure.
  - product exhibits high failure rate.
  - product difficult to understand.

Which software requires it?

- Most old and legacy applications
- Very large applications being used since long whose maintainability has decreased because of rampant patching.

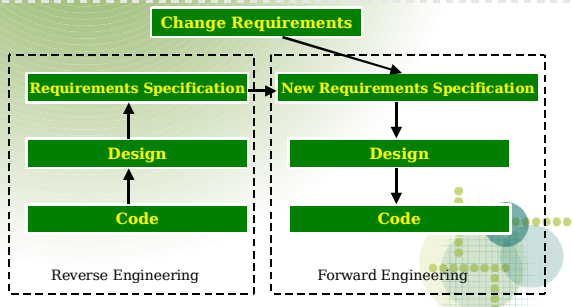
Steps >

- Step 1 Reverse Engineering
  - The old code is analyzed (abstracted) to extract the module specifications.
  - The module specifications are analyzed to produce the design.
  - The design is analyzed (abstracted) to produce the original requirements specification.

Steps

- Step 2 Preparation of New SRS
  - The change requests are then applied to the requirements specification to arrive at the new requirements specification.
- Step 3 Forward Engineering
  - Done by adopting software engineering principles rigorously

Re-engineering process model



Advantages

- produces better design than the original product,
- produces required documents,
- often results in higher efficiency.
- Efficiency improvements are brought about by better design.
- However, this approach is more costly than ordinary maintenance.

Finally,

There

Is

no

End