

Chapter 36 - Software Maintenance

Software Maintenance can Maintenance in

- Software is released to end-users, and
 - within days, bug reports filter back to the software engineering organization.
 - within weeks, one class of users indicates that the software must be changed so that it can accommodate the special needs of their environment.
 - within months, another corporate group who wanted nothing to do with the software when it was released, now recognizes that it may provide them with unexpected benefit. They'll need a few enhancements to make it work in their world.

Maintainable Software

- Maintainable software exhibits effective modularity
- It makes use of design patterns that allow ease of understanding.
- It has been constructed using well-defined coding standards and conventions, leading to source code that is self-documenting and understandable.
- It has undergone a variety of quality assurance techniques that have uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not be around when changes must be made.

Software Supportability

- the capability of supporting a software system over its whole product life.
- The software should contain facilities to assist support personnel when a defect is encountered in the operational environment (and make no mistake, defects *will* be encountered).
- Support personnel should have access to a database that contains records of all defects that have already been encountered—their characteristics, cause, and cure.

Reengineering

- 1- Business processes
- 2- IT system
- 3- Software application

Business Process Reengineering

- **Business definition.** Business goals are identified within the context of four key drivers: cost reduction, time reduction, quality improvement, and personnel development and empowerment.
- **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified.
- **Process evaluation.** The existing process is thoroughly analyzed and measured.
- **Process specification and design.** Based on information obtained during the first three BPR activities, use-cases are prepared for each process that is to be redesigned.
- **Prototyping.** A redesigned business process must be prototyped before it is fully integrated into the business.

- **Refinement and instantiation.** Based on feedback from the prototype, the business process is refined and then instantiated within a business system.

BPR Principles

- Organize around outcomes, not tasks.
- Have those who use the output of the process perform the process.
- Incorporate information processing work into the real work that produces the raw information.
- Treat geographically dispersed resources as though they were centralized.
- Link parallel activities instead of integrated their results. When different
- Put the decision point where the work is performed, and build control into the process.
- Capture data once, at its source.

Document Restructuring

- *Creating documentation is far too time consuming.* If the system works, we'll live with what we have. In some cases, this is the correct approach.
- *Documentation must be updated, but we have limited resources.* We'll use a "document when touched" approach. It may not be necessary to fully redocument an application.
- *The system is business critical and must be fully redocumented.* Even in this case, an intelligent approach is to pare documentation to an essential minimum

Code Restructuring

- Source code is analyzed using a restructuring tool.
- Poorly design code segments are redesigned
- Violations of structured programming constructs are noted and code is then restructured (this can be done automatically)
- The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced
- Internal code documentation is updated.

Forward Engineering

- 1.The cost to maintain one line of source code may be 20 to 40 times the cost of initial development of that line.
- 2.Redesign of the software architecture (program and/or data structure), using modern design concepts, can greatly facilitate future maintenance.
- 3.Because a prototype of the software already exists, development productivity should be much higher than average.
- 4.The user now has experience with the software. Therefore, new requirements and the direction of change can be ascertained with greater ease.

5. CASE tools for reengineering will automate some parts of the job.

6. A complete software configuration (documents, programs and data) will exist upon completion of preventive maintenance.