Maintenance Phase

• Any change to any component of the product after it has passed the acceptance test
  – Requirements
  – Specification
  – Design
  – Implementation
  – Documentation
    • Manuals

Maintenance Type

• Corrective Maintenance
  – To correct residual faults
    • Specification, design, implementation, documentation, or any other types of faults
  – On average, 17.5% of maintenance
Maintenance Type

• Perfective Maintenance
  – Client requests changes to improve product effectiveness
    • Add additional functionality
    • Make product run faster
    • Improve maintainability
  – On average, 60.5% of maintenance

• Adaptive Maintenance
  – Responses to changes in environment in which product operates
    • Product ported to new compiler, operating system, and/or hardware
    • Change to tax code
    • 9-digit ZIP codes
  – On average, 18% of maintenance
Maintenance Difficulties

• About 67% of the total cost of a product accrues during the maintenance phase
• Maintenance is a major income source
• Nevertheless, even today many organizations assign maintenance to
  – Unsupervised beginners
  – Less competent programmers

What is Required of Maintenance Programmers?

• Maintenance is one of the most difficult aspects of software production since it incorporates aspects of every phases
• Suppose a fault report is handed to a maintenance programmer
  – The programmer must have following abilities:
    • Investigative
    • Debugging
    • Design
    • Implementation
    • Integration
    • Testing
What is Required of Maintenance Programmers?

• Tools used by maintenance programmer to find the fault:
  – The fault report filed by user
  – The source code
  – And often nothing else

• Maintenance programmer must have superb debugging skills
  – The fault could lie anywhere within the product
  – The original cause of the fault might lie in the non-existent specifications or design documents

Corrective Maintenance

• Suppose that the maintenance programmer has located the fault
  – Problem is: how to fix it without introducing a regression fault

• How to minimize regression faults
  – Consult the detailed documentation for product as a whole
  – Consult the detailed documentation for each individual module
Corrective Maintenance

• What usually happens
  – There is no documentation at all
  – The documentation is incomplete
  – The documentation is faulty
• The programmer must deduce from the source code all the information needed to avoid introducing a regression fault

Corrective Maintenance

• Programmer changes the source code
  – Test that the modification works correctly
  – Check for regression faults by using stored test data
    • Add specially constructed test cases to stored test data for future regression testing
• Document all changes
• Major skills required for corrective maintenance
  – Superb diagnostic skills
  – Superb testing skills
  – Superb documentation skills
Adaptive and Perfective Maintenance

• The maintenance programmer must go through the phases of
  – Requirements
  – Specifications
  – Design
  – Implementation and integration

• Using the existing product as a starting point

• No form of maintenance
  – Is a task for an unsupervised beginner
  – Should be done by a less skilled computer professional

Maintenance Reword

• Maintenance is a thankless task in every way
  – Maintainers deal with dissatisfied users
  – If the user were happy, the product would not need maintenance
  – The user’s problems are often caused by the individuals who developed the product, not the maintainer
  – The code itself may be badly written
  – Maintenance is despised by many software developers
  – Unless good maintenance service is provided, the client will take future development business elsewhere
  – Maintenance is the most important phase of software production, the most difficult—and most thankless
Maintenance Reword

• How can this situation be changed?
  – Managers must assign maintenance to their best software engineers
  – Pay software engineers accordingly

Management of Maintenance

• A mechanism is needed for changing a product
  – If the product appears to function incorrectly, the user files a fault report
    • It must include enough information to enable the maintenance programmer to recreate the problem
  – Ideally, every fault should be fixed immediately
    • In practice, immediate preliminary investigation
• The maintenance programmer should first consult
  – The fault report
  – All reported faults not yet fixed
  – Suggestions for working around them
Ensuring Maintainability

• Maintenance is not a one-time effort
• We must plan for maintenance over the entire life cycle
  – Design phase—use information-hiding techniques
  – Implementation phase—select variable names meaningful to future maintenance programmers
  – Documentation must be complete and correct, and reflect current version of every module
• During the maintenance phase, maintainability must not be compromised
  – Always be conscious of the inevitable further maintenance
• Principles leading to maintainability are equally applicable to the maintenance phase itself

Maintenance of Object-Oriented Software

• The object-oriented paradigm promotes maintenance
  – The product consists of independent units
  – Encapsulation (conceptual independence)
  – Information hiding (physical independence)
  – Message-passing is the sole communication
• In reality there are three obstacles
In reality there are three obstacles

• The complete inheritance hierarchy can be large
  – Problem
    • Inheritance tree may be spread over the entire product
  – Solution
    • A CASE tool can help developer to understand the inheritance tree

In reality there are three obstacles

• The consequences of polymorphism and dynamic binding
  – Product fails on the invocation `myFile.open()`
  – Which version of `open` contains the fault?
    • A CASE tool cannot help (static tool)
    • We must trace
In reality there are three obstacles

- The consequences of inheritance
  - Create new subclass by inheritance
    - Does not affect superclass
    - Does not affect any other subclass
  - Modify this new subclass
    - Again, no affect
  - Modify a superclass
    - All descendent subclasses are affected
  - Inheritance can have
    - Positive effect on development
    - Negative effect on maintenance

Maintenance versus Development Skills

- Skills needed for maintenance include
  - Ability to determine cause of failure of large product
    - Also needed during integration and product testing
  - Ability to function effectively without adequate documentation
    - Documentation rarely complete until delivery
- Skills in specification, design, implementation, testing
  - All four activities carried out during development
- Skills needed for maintenance same as those for other phases
Maintenance versus Development Skills

- **Key Point**
  - Maintenance programmers must not merely be skilled in broad variety of areas, they must be **highly** skilled in all those areas
  - Specialization impossible for the maintenance programmer
- **Maintenance is the same as development, only more!!**

Reverse Engineering

- **Reengineering**
  - Reverse engineering, followed by forward engineering
  - Lower to higher to lower levels of abstraction
- **Restructuring**
  - Improving product without changing functionality
  - Prettyprinting, structuring code, improving maintainability
- **When the only documentation is the code itself**
  - Start with the code
  - Recreate the design
  - Recreate the specifications (extremely hard)
  - CASE tools help (flowcharters, other visual aids)
- **What if we have only the executable code?**
  - Treat as black box
Testing during the Maintenance Phase

- Maintainers view a product as loosely related modules
  - developers were not involved in development of the product
- Regression testing is essential
  - Store test cases and outcomes, modify as needed

CASE Tools for the Maintenance Phase

- Version, revision control tools
- Reengineering tools
  - Battlemap, Teamwork, Bachman Product Set
Metrics for the Maintenance Phase

• Metrics for development phases
• Fault report metrics

Challenges of the Maintenance Phase

• The development-then-maintenance model is unrealistic today
  – The client’s requirements frequently change before the product is delivered
  – Faults often have to be fixed before the product is delivered
  – Development from scratch is almost unknown today.
    • Instead, products are built from reused components.
• Products are modified before delivery