OBJECT-ORIENTED AND CLASSICAL SOFTWARE ENGINEERING

EIGHTH EDITION, WCB/MCGRAW-HILL, 2011

STEPHEN R. SCHACH

Edited by Hyungbae Park

University of Central Missouri

CHAPTER 16

POSTDELIVERY MAINTENANCE

Overview

- Why postdelivery maintenance is necessary
- What is required of postdelivery maintenance programmers?
- Postdelivery maintenance mini case study
- Management of postdelivery maintenance
- Postdelivery maintenance skills versus development skills
- Reverse engineering
- Testing during postdelivery maintenance
- Challenges of postdelivery maintenance

Postdelivery Maintenance

- Postdelivery maintenance
  - **_Any_ change to _any_ component of the product (including documentation) after it has passed the acceptance test**

16.1 Why Postdelivery Maintenance Is Necessary

1. Corrective maintenance
2. Perfective maintenance
3. Adaptive maintenance

Why Postdelivery Maintenance Is Necessary (contd)

- Corrective maintenance
  - To correct residual faults
    - Analysis, design, implementation, documentation, or any other type of faults

Why Postdelivery Maint. Is Necessary (contd)

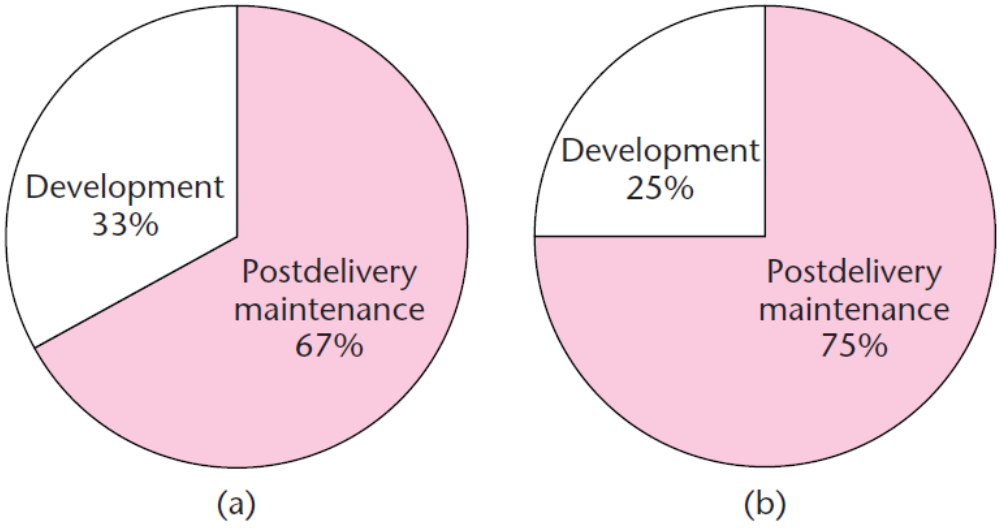
- Perfective maintenance
  - Client requests changes to improve product effectiveness
    - Add additional functionality
    - Make product run faster
    - Improve maintainability

Why Postdelivery Maint. Is Necessary (contd)

- Adaptive maintenance
  - Responses to changes in the *environment* in which the product operates
    - The product is ported to a new compiler, operating system, and/or hardware
    - A change to the tax code
    - 9-digit ZIP codes

16.2 What Is Required of Postdelivery Maintenance Programmers?

- During the software life cycle, more time is spent on postdelivery maintenance than on any other activity
  - At least 67% of the total cost of a product accrues during postdelivery maintenance



- Approximate average cost percentages of development and postdelivery maintenance (a) between 1976 and 1981 a) between 1992 and 1998.

16.2 What Is Required of Postdelivery Maintenance Programmers?

- Maintenance is a major income source
- Nevertheless, even today many organizations assign maintenance to
  - Unsupervised beginners, and
  - Less competent programmers

What is Required of Postd. Maint. Prog. (contd)?

- Postdelivery maintenance is one of the most difficult aspects of software production because
  - Postdelivery maintenance incorporates aspects of all other workflows

What is Required of Postd. Maint. Prog. (contd)?

- Suppose a defect report is handed to a maintenance programmer
  - Recall that a “defect” is a generic term for a fault, failure, or error

Corrective Maintenance

- What tools does the maintenance programmer have to find the fault?
  - The defect report filed by user
  - The source code
  - And often nothing else

Corrective Maintenance (contd)

- A maintenance programmer must therefore have far above debugging skills
  - The fault could lie anywhere within the product
  - The original cause of the fault might lie in the by now non-existent specifications or design documents

Corrective Maintenance (contd)

- Suppose that the maintenance programmer has located the fault
- Problem:
  - How to fix it without introducing a **regression fault**

Corrective Maintenance (contd)

- How to minimize regression faults
  - Consult the detailed documentation for the product as a whole
  - Consult the detailed documentation for each individual module
- What usually happens
  - There is no documentation at all, or
  - The documentation is incomplete, or
  - The documentation is faulty

Corrective Maintenance (contd)

- The programmer must deduce from the source code itself all the information needed to avoid introducing a regression fault

- The programmer now changes the source code

The Programmer Now Must

- Test that the modification works correctly
  - Using specially constructed test cases
- Check for regression faults
  - Using stored test data
- Add the specially constructed test cases to the stored test data for future **regression testing**
- Document all changes

Regression testing

- Once the programmer has determined that the desired changes have been implemented, the product must be tested against previous test cases
  - To make certain that the functionality of the rest of the product has not been compromised

- This procedure is called **regression testing**
  - To assist in regression testing, it is necessary that all previous test cases be retained, together with the results of running those test cases

Corrective Maintenance (contd)

- Major skills are required for corrective maintenance
  - Superb diagnostic skills
  - Superb testing skills
  - Superb documentation skills

Adaptive and Perfective Maintenance

- The maintenance programmer must go through the
  - Requirements
  - Specifications
  - Design
  - Implementation and integration

  workflows, using the existing product as a starting point

Adaptive and Perfective Maintenance (contd)

- When programs are developed
  - Specifications are produced by analysis experts
  - Designs are produced by design experts
  - Code is produced by programming experts

- But a maintenance programmer must be expert in all three areas, and also in
  - Testing, and
  - Documentation

Conclusion

- No form of maintenance
  - Is a task for an unsupervised beginner, or
  - Should be done by a less skilled computer professional

The Rewards of Maintenance

- 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
  - Postdelivery maintenance is despised by many software developers
  - Unless good maintenance service is provided, the client will take future development business elsewhere
  - Postdelivery maintenance is the most challenging aspect of software production — and the most thankless

The Rewards of Maintenance (contd)

- How can this situation be changed?
  - Managers must assign maintenance to their best programmers (top computer professionals), and
  - Pay them accordingly

16.3 Postdelivery Maintenance Mini Case Study

- In countries with centralized economies, the government controls the distribution and marketing of agricultural products
- In one such country, temperate fruits, such as peaches, apples, and pears, were the responsibility of the Temperate Fruit Committee (TFC)
- One day, the chairman of the TFC asked a government computer consultant to computerize the operations of the TFC
- The chairman informed the consultant that there are exactly seven temperate fruits:
  - Apples, apricots, cherries, nectarines, peaches, pears, and plums

Postdelivery Maintenance Mini Case Study (contd)

- The database was to be designed for those seven fruits, no more and no less
  - After all, that was the way that the world was, and the consultant was not to waste time and money allowing for any sort of expandability
- The product was duly delivered to the TFC
- About a year later, the chairman summoned the maintenance programmer responsible for the product
  - "It seems that kiwi fruit is a temperate fruit that has just started to be grown in our country, and the TFC is responsible for it. Please change the product accordingly."
- The consultant had provided a number of unused fields in the relevant database records
  - By slightly rearranging certain items, the maintenance programmer was able to incorporate kiwi fruit, the eighth temperate fruit, into the product

Postdelivery Maintenance Mini Case Study (contd)

- Another year went by, and the product functioned well
- His committee was now responsible for all fruit produced in that country, not just temperate fruit,
  - So the product now had to be modified to incorporate the 26 additional kinds of fruit on the list he handed to the maintenance programmer
- The programmer protested, pointing out that this change would take almost as long as rewriting the product from scratch.

  "Nonsense," replied the chairman.

  "You had no trouble adding kiwi fruit. Just do the same thing another 26 times!"

Postdelivery Maintenance Mini Case Study (contd)

- Lessons to be learnt from this
  - The problem was caused by the developer, not the maintainer
  - A maintainer is often responsible for fixing other people's mistakes
  - The client frequently does not understand that postdelivery maintenance can be difficult, or all but impossible
  - This is exacerbated when previous apparently similar perfective and adaptive maintenance tasks have been carried out
  - All software development activities must be performed with an eye on future postdelivery maintenance

16.5 Management of Postdelivery Maintenance

- Various issues regarding management of postdelivery maintenance are now considered

1. Defect reports
2. Authorizing Changes to the Product
3. Ensuring Maintainability
4. Problem of Repeated Maintenance

16.5 .1 Defect Reports

- We need a mechanism for changing a product

- If the product appears to function incorrectly, the user files a **defect report**
  - It must include enough information **to enable the maintenance programmer to recreate the problem**
  - The maintenance programmer must indicate the severity of the defect
    - Typical severity categories include critical, major, normal, minor, and trivial

Defect Reports (contd)

- The maintenance programmer should first consult the **defect report** file

- It contains
  - **All reported defects not yet fixed,** and
  - **Suggestions for working around them**

Defect Reports (contd)

- If the defect has been previously reported
  - Give the information in the defect report file to the user
- If it is a new defect
  - The maintenance programmer should try to find
    - The cause,
    - A way to fix it, and
    - A way to work around the problem
  - The new defect is now filed in the defect report file, together with supporting documentation
    - Listings
    - Designs
    - Manuals

Defect Reports (contd)

- If it is a new defect (contd)
  - The file should **also contain the client's requests for perfective and adaptive maintenance**
    - The contents of the file must be prioritized by the client
    - The next modification is the one with the highest priority
  - Copies of defect reports must be circulated to all
    - Including: An estimate of when the defect can be fixed

Defect Reports (contd)

- In an ideal world
  - We fix every defect immediately
  - Then we distribute the new version of the product to all the sites
- In the real world
  - We distribute defect reports to all sites
  - We do not have the staff for instant maintenance
  - It is cheaper to make a number of changes at the same time, particularly if there are multiple sites

16.5.2 Authorizing Changes to the Product

- Corrective maintenance
  - Assign a maintenance programmer to determine the fault and its cause, then repair it
  - Test the fix, test the product as a whole (regression testing)
  - Update the documentation to reflect the changes made
  - Update the prologue comments to reflect
    - What was changed,
    - Why it was changed,
    - By whom, and
    - When

Authorizing Changes to the Product (contd)

- Adaptive and perfective maintenance
  - As with corrective maintenance, except there is no defect report
  - There is a change in requirements instead

Authorizing Changes to the Product (contd)

- What if the programmer has not tested the fix adequately?
  - Before the product is distributed, it must be tested by the SQA group
- Testing is difficult and time consuming
  - Performed by the SQA group

16.5.3 Ensuring Maintainability

- Maintenance is not a one-time effort

- We must plan for maintenance over the entire life cycle
  - Design workflow — use information-hiding techniques
  - Implementation workflow — select variable names meaningful to future maintenance programmers
  - Documentation must be complete and correct, and reflect the current version of every artifact

16.5.4 The Problem of Repeated Maintenance

- The moving target problem is frustrating to the development team
- Frequent changes have an adverse effect on the maintainability of the product

The Problem of Repeated Maintenance (contd)

- The moving target problem
  - The problem is exacerbated during postdelivery maintenance
  - The more changes there are
    - The more the product deviates from its original design
    - The more difficult further changes become
    - Documentation becomes even less reliable than usual
    - Regression testing files are not up to date
    - A total rewrite may be needed for further maintenance

The Problem of Repeated Maintenance (contd)

- The moving target problem (contd)
  - Apparent solution
    - Freeze the specifications once they have been signed off until delivery of the product
    - After each request for perfective maintenance, freeze the specifications for (say) 3 months or 1 year
  - In practice
    - The client can order changes the next day
    - If willing to pay the price, the client can order changes on a daily basis

The Problem of Repeated Maintenance (contd)

- Warning
  - It is no use implementing changes slowly
  - The relevant personnel are replaced
  - Nothing can be done if the person calling for repeated change has sufficient clout

16.6 Maintenance of OO Software

- The object-oriented paradigm apparently promotes maintenance in four ways
  - The product consists of independent units
  - Encapsulation (conceptual independence)
  - Information hiding (physical independence)
  - Message-passing is the sole communication

- The reality is somewhat different

Maintenance of OO Software (contd)

- Three obstacles
  - The complete inheritance hierarchy can be large
  - The consequences of polymorphism and dynamic binding
  - The consequences of inheritance

The complete inheritance hierarchy can be large


```
class UndirectedTreeClass
{
   ...
   void displayNode (Node a);
   ...
}// class UndirectedTreeClass

class DirectedTreeClass : public UndirectedTreeClass
{

   ...
}// class DirectedTreeClass

class RootedTreeClass : public DirectedTreeClass
{

   ...
   void displayNode (Node a);
   ...
}// class RootedTreeClass

class BinaryTreeClass : public RootedTreeClass
{

   ...
}// class BinaryTreeClass

class BalancedBinaryTreeClass : public BinaryTreeClass
{
   Node          hhh;
   displayNode (hhh);
}// class BalancedBinaryTreeClass
```

Figure 14.1

The complete inheritance hierarchy can be large (contd)

- To find out what `displayNode()` does in **`BalancedBinaryTreeClass`**, a maintenance programmer must scan the complete tree
  - The inheritance tree may be spread over the entire product
  - A far cry from "independent units"
- Solution
  - A CASE tool can flatten the inheritance tree

Polymorphism and Dynamic Binding

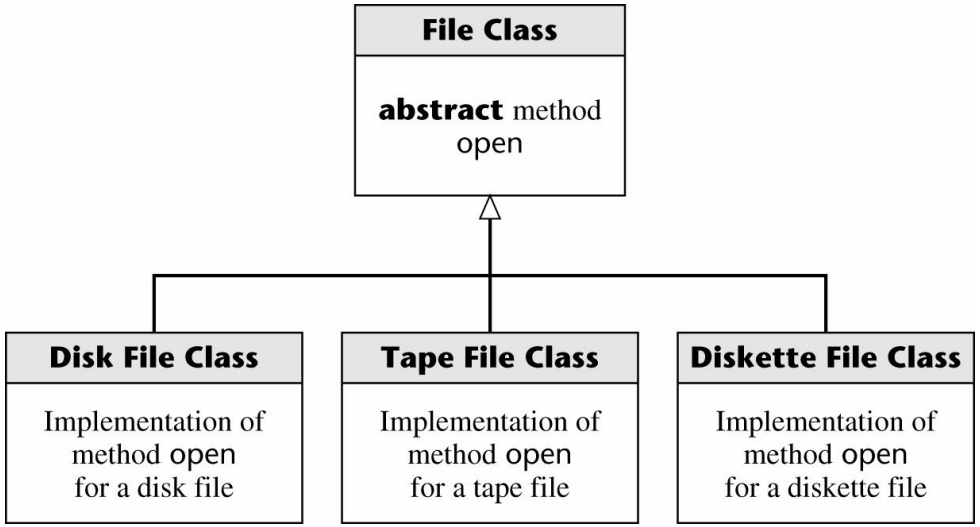


Figure 14.2

- The product fails on the invocation `myFile.open()`
- Which version of `open` contains the fault?
  - A CASE tool cannot help (static tool)
  - We must trace

Polymorphism and Dynamic Binding (contd)

- Polymorphism and dynamic binding can have
  - A positive effect on development, but
  - A negative effect on maintenance

Consequence of Inheritance

- Create a new subclass via inheritance
- The new subclass
  - Does not affect any superclass, and
  - Does not affect any other subclass
- Modify this new subclass
  - Again, no affect
- Modify a superclass
  - All descendent subclasses are affected
  - “Fragile base class problem”

Consequence of Inheritance (contd)

- Inheritance can have
  - A positive effect on development, but
  - A negative effect on maintenance

16.7 Postdelivery Maintenance versus Development Skills

- The skills needed for maintenance include
  - The ability to determine the cause of failure of a large product
    - Also needed during integration and product testing
  - The ability to function effectively without adequate documentation
    - Documentation is rarely complete until delivery
  - Skills in analysis, design, implementation, and testing
    - All four activities are carried out during development

Postdelivery Maintenance vs. Development Skills (contd)

- The skills needed for postdelivery maintenance are the same as those for the other workflows

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

- Postdelivery maintenance is the same as development, only more so

16.8 Reverse Engineering

- When the only documentation for postdelivery maintenance is the code itself
  - Start with the code
  - Recreate the design
  - Recreate the specifications (extremely hard)
  - CASE tools can help (flowcharters, other visual aids)

- Forward engineering
  - Within the context of reverse engineering, the usual development process that proceeds from analysis through design to implementation is called **forward engineering**

Reverse Engineering (contd)

- Restructuring
  - **Improving the product without changing its functionality**
  - Examples:
    - Pretty printer - display the code more clearly
    - Unstructured → Structured code
    - Improving maintainability
    - Refactoring

Reverse Engineering (contd)

- What if we have only the executable code?
  - Worse situation – if the source code is lost
  - Treat the product as a black box
  - Deduce the specifications from the behavior of the current product

16.9 Testing during Postdelivery Maintenance

- Maintainers tend to view a product as a set of loosely related components
  - They were not involved in the development of the product
  - Generally is not aware that a change to one code artifact may seriously affect one or more other artifacts and hence the product as a whole

- Regression testing is essential
  - Testing the changed product against previous test cases to ensure that it still works correctly
  - For this reason, it is vital to store all test cases, together with their expected outcomes, in machine-readable form
  - Store test cases and their outcomes, modify as needed

16.13 Challenges of Postdelivery Maintenance

- The chapter describes numerous challenges

- The hardest challenge to solve
  - Maintenance is harder than development, but
  - Developers tend to look down maintainers, and
  - Are frequently paid more