September 1, 2002
to do:

Include case study througout the book and the larger example at the end.

Lecture notes

Topic SM
Maintenance

DAAD project
"Joint Course on Software Engineering"

Humboldt University of Berlin, University of Novi Sad, University of Plovdiv, University of Skopje, University of Belgrade, University of Niš, University of Kragujevac

Parts of this Chapter use material from the textbook
H. Balzert, "Software-Technik", Vol. 1, 2nd ed., Spektrum Akademischer Verlag, 2001

DocID: D020521-E020530

Literature

- Braude: Software engineering – an object-oriented perspective, Wiley, 2000.

SM. Maintenance

a) Activities and issues of software maintenance
b) Types of software maintenance
c) Maintenance techniques
d) Standardization
e) Management of maintenance
d) Metrics

Software maintenance

- The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [IEEE 610].
- 40% - 90% of the total life cycle costs is for maintenance.
- Maintenance does not normally involve major changes to the system's architecture
- Changes are implemented by modifying existing components and adding new components to the system
- Example: the year 2000 problem.

Maintenance is inevitable

- The system requirements are likely to change while the system is being developed because the environment is changing. Therefore a delivered system won't meet its requirements!
- Systems are tightly coupled with their environment. When a system is installed in an environment it changes that environment and therefore changes the system requirements.
- Systems MUST be maintained therefore if they are to remain useful in an environment

Servicing maintenance request - 1

- **Be prepared to keep required metrics. Include..**
  - **… lines of code added**
  - **… lines of code changed**
  - **… time taken: 1. preparation 2. design 3. code 4. test**
- **Ensure that the request has been approved**
- **Understand the problem thoroughly**
  - **reproduce the problem**
    - **otherwise get clarification**
- **Classify the MR as *repair* or *enhancement***
- **Decide whether the implementation requires a redesign at a higher level**
  - **if so, consider batching with other MR's**
- **Design the required modification**
  (i.e., incorporate the change)

Servicing maintenance request - 2

- Plan transition from current design
- Assess change's impact throughout the application
  - small changes can have major impact!
- Implement the changes
- Perform unit testing on the changed parts
- Perform regression testing
  - ensure changes haven't compromised existing capabilities
- Perform system testing with new capabilities
- Update the configuration, requirement, design and test documentation

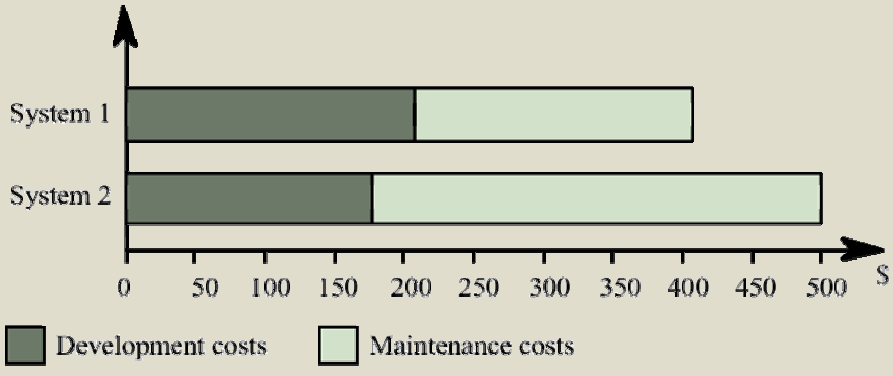
Issues

- *Management*
  - Return on investment hard to define
- *Process*
  - Extensive coordination required to handle stream of Maintenance Requests
- *Technical*
  - Covering full impact of changes
  - Testing very expensive compared with the utility of each change
    - focused tests ideal but expensive
    - regression testing still required

Maintenance costs

- Usually greater than development costs (2* to 100* depending on the application)
- Affected by both technical and non-technical factors
- Increases as software is maintained. Maintenance corrupts the software structure so makes further maintenance more difficult.
- Ageing software can have high support costs (e.g. old languages, compilers etc.)

Development/maintenance costs



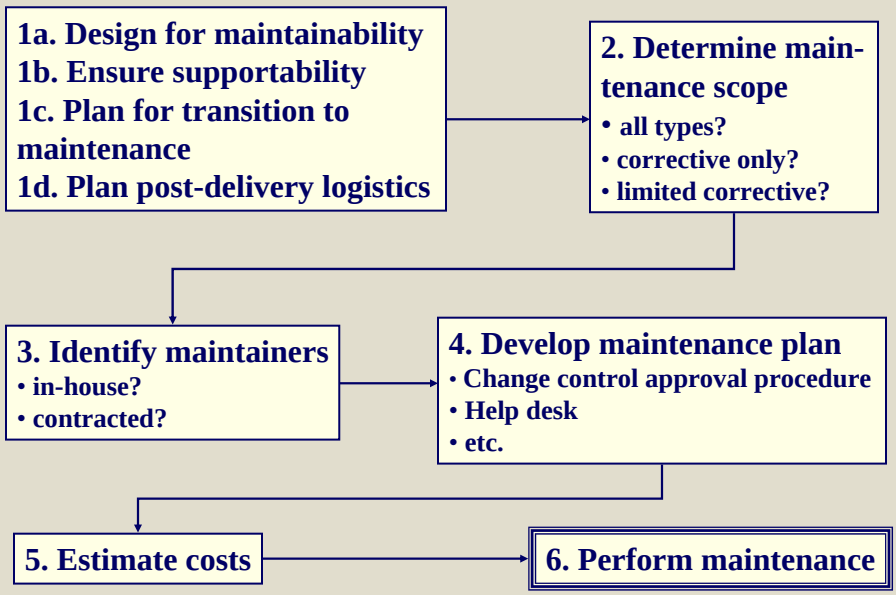
Maintenance cost factors

- Team stability
  - Maintenance costs are reduced if the same staff are involved with them for some time
- Contractual responsibility
  - The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change
- Staff skills
  - Maintenance staff are often inexperienced and have limited domain knowledge
- Program age and structure
  - As programs age, their structure is degraded and they become harder to understand and change

Example: Estimating the Cost of Servicing a Maintenance Request

Activity	Estimate (person-days)		Activity	Estimate (person-days)
1. Understand the problem and identify the functions that must be modified or added.	2 - 5		6. Compile and integrate into baseline	2 - 3
2. Design the changes	1 - 4		7. Test functionality of changes	2 - 4
3. Perform impact analysis	1 - 4		8. Perform regression testing	2 - 4
4. Implement changes in source code	1 - 4		9. Release new baseline and report results	1
5. Change SRS, SDD, STP, configuration status	2 - 6		**TOTAL**	**14 - 35**

RoadMap to Establish Maintenance



SM. Maintenance

a) Activities and issues of software maintenance
b) Types of software maintenance
c) Maintenance techniques
d) Standardization
e) Management of maintenance
d) Metrics

Types of maintenance

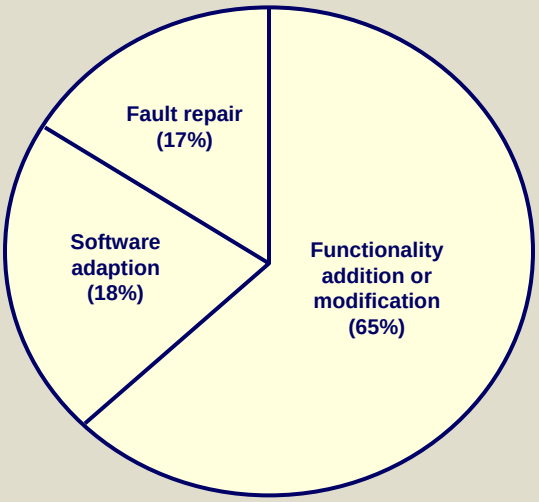
Fixing

- **Corrective**
  - defect identification and removal
- **Adaptive**
  - changes resulting from operating system, hardware or DBMS changes

Enhancing

- **Perfective**
  - changes resulting from user requests
- **Preventative**
  - changes made to the software to make it more maintainable

Distribution of maintenance effort



SM. Maintenance

a) Activities and issues of software maintenance
b) Types of software maintenance
c) Maintenance techniques
d) Standardization
e) Management of maintenance
d) Metrics

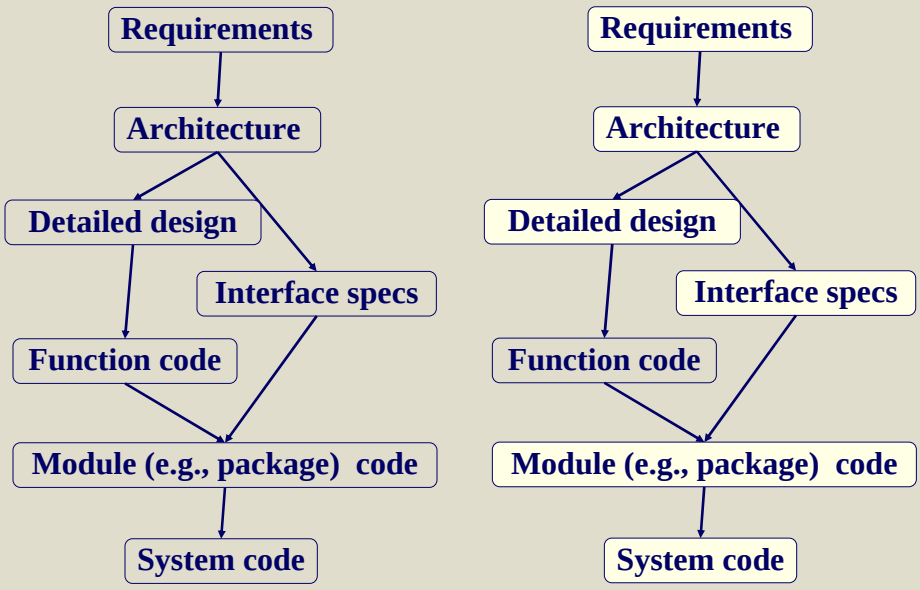
Maintenance techniques

- Impact analysis
- *Reverse engineering*
- Reengineering
  - Refactoring
- Legacy applications
- Updating documentation

Impact analysis

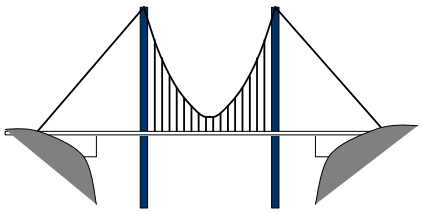
- 19% of defects emanated from requirements phase, 52% from design, 7% from programming [Weiss].

Possible impacts - corrections

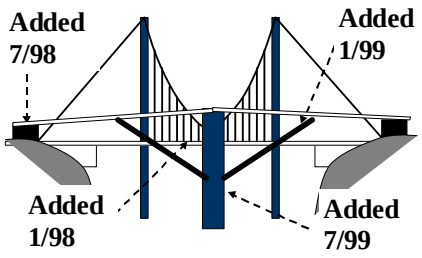


Maintenance without reengineering

The Beginning Product

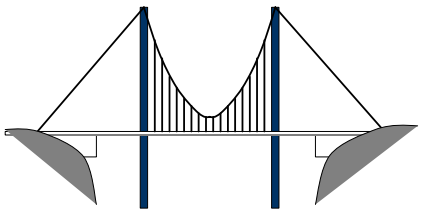


Expansion Without Reengineering

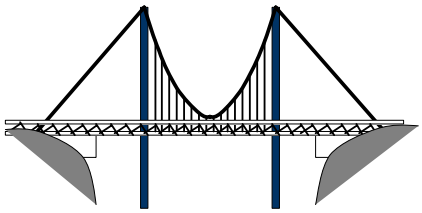


Maintenance with reengineering

The Beginning Product



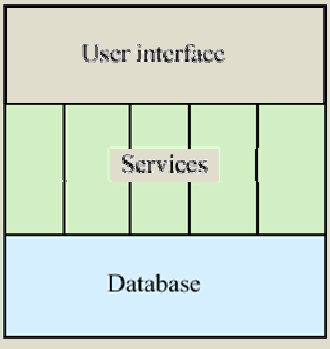
Reengineered Expansion



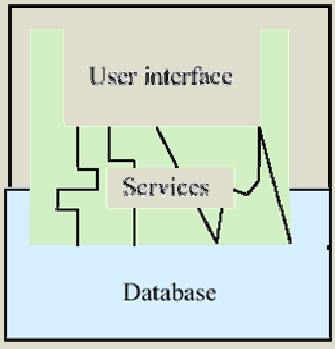
Legacy system structure

- Ideally, for distribution, there should be a clear separation between the user interface, the system services and the system data management
- In practice, these are usually intermingled in older legacy systems

Legacy system structure (2)



Ideal model for distribution

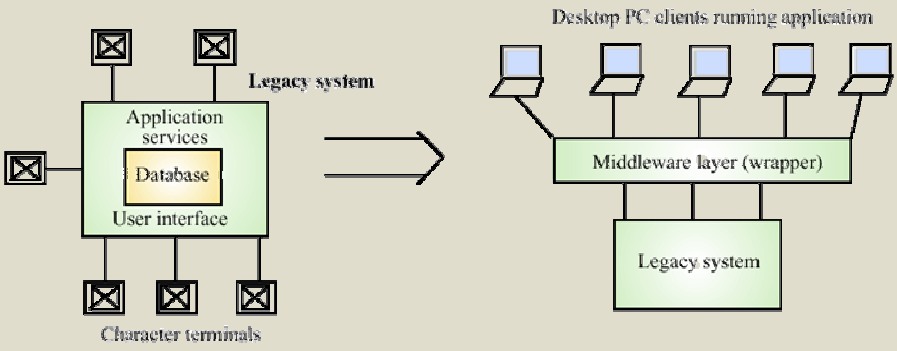


Real legacy systems

Legacy applications

- Continue to *maintain*
- Discontinue maintenance and --
  1. *Replace*
     - buy replacement
     - OR build replacement
       - reverse engineer legacy system
       - or develop new architecture
       - possibly replace in phases
  2. *Incorporate* as integral part of new application
     - freeze maintenance
  3. *Encapsulate* and use as server
     - consider using *Adapter* design pattern

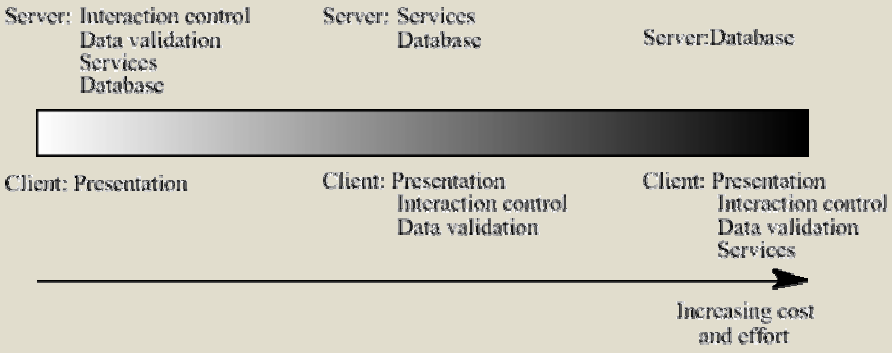
Legacy system distribution



Distribution options

- The more that is distributed from the server to the client, the higher the costs of architectural evolution
- The simplest distribution model is UI distribution where only the user interface is implemented on the server
- The most complex option is where the server simply provides data management and application services are implemented on the client

Distribution option spectrum



22. Maintenance

a) Activities and issues of software maintenance
b) Types of software maintenance
c) Maintenance techniques
d) Standardization
e) Management of maintenance
d) Metrics

IEEE Standard 840-1994

1. Problem identification
1.1 Input 1.2 Process
1.3 Control 1.4 Output
1.5 Quality factors
1.6 Metrics

2. Analysis
2.1 Input
2.2 Process
2.2.1 Feasibility analysis
2.2.2 Detailed analysis
2.3-2.6 Control, Output, Quality factors, Metrics.

3. Design
3.1-3.6 Input, Process, Control, Output, Quality factors, Metrics.

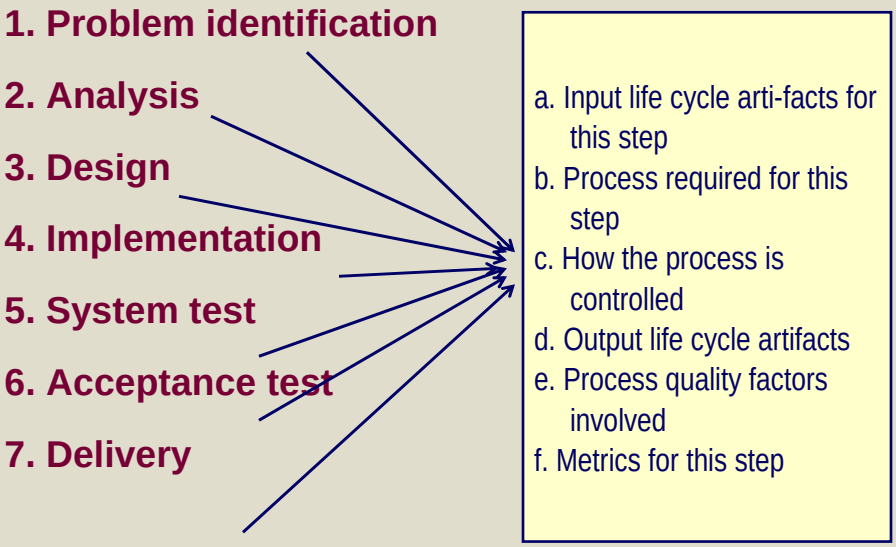
4. Implementation
4.1 Input
4.2 Process
4.2.1 Coding and & testing
4.2.3 Risk analysis & review
4.2.4 Test-readiness review
4.3-4.6 Control, Output, Quality factors, Metrics.

5. System test
5.1-5.6 Input, Process, Control, Output, Quality factors, Metrics.

6. Acceptance test
6.1-6.6 Input, Process, Control, Output, Quality factors, Metrics.

7. Delivery
7.1-7.6 Input, Process, Control, Output, Quality factors, Metrics.

Five Attributes of Each Maintenance Step (IEEE)

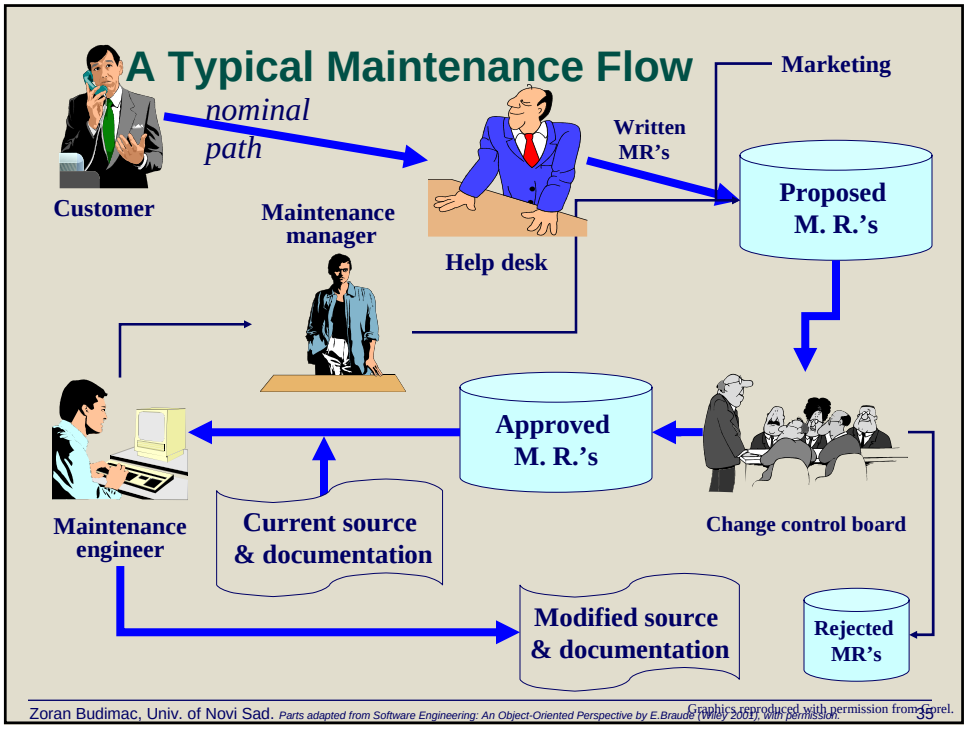


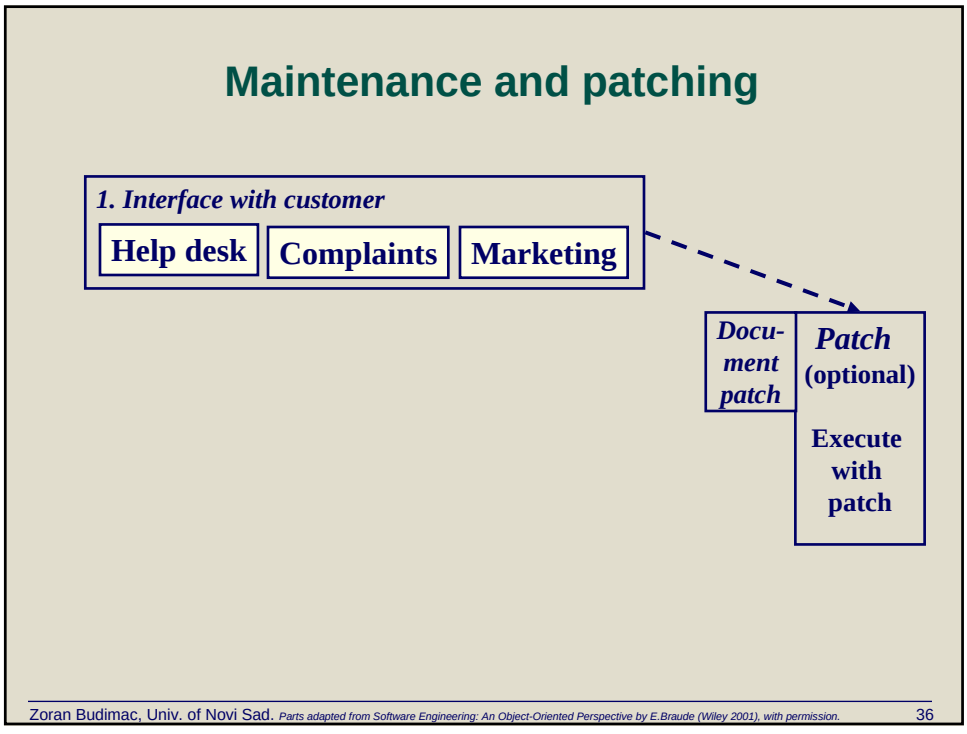
	IEEE 1219-1992 **Maintenance phase 1: *Problem Identification***
a. Input	•The Maintenance Request (MR)
b. Process	•Assign change number •Classify by type and severity etc. •Accept or reject change •Make preliminary cost estimate •Prioritize
c. Control	•Identify MR uniquely •Enter MR into repository
d. Output	•Validated MR
e. Selected quality factors	•Clarity of the MR •Correctness of the MR (e.g., type)
f. Selected metrics	•Number of omissions in the MR •Number of MR submissions to date •Number of duplicate MR's •Time expected to confirm the problem

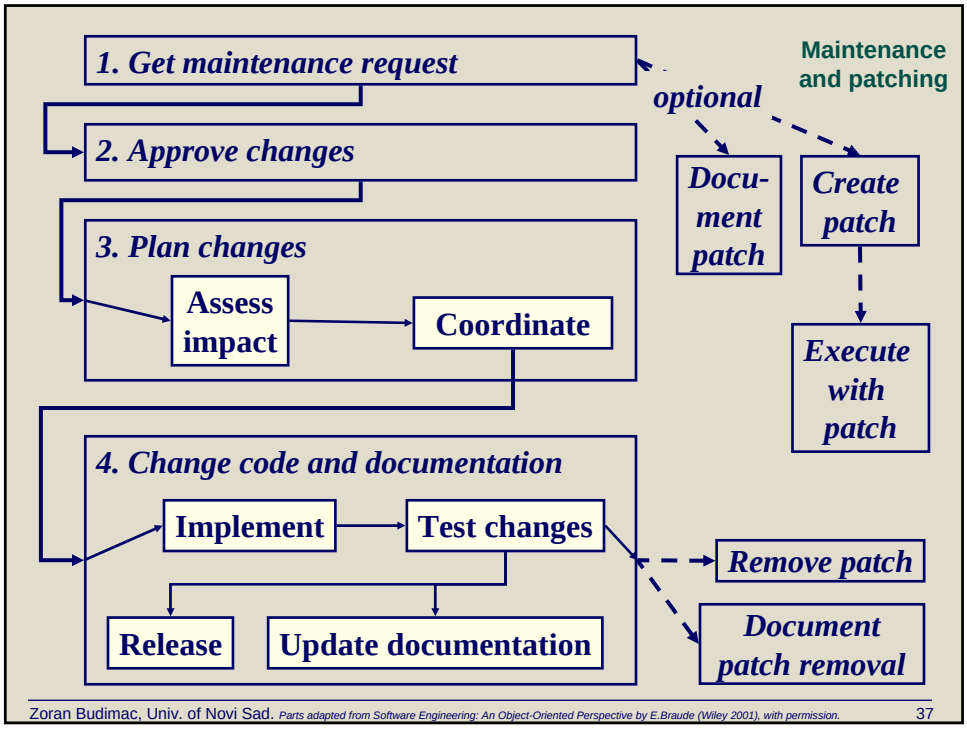
	IEEE 1219-1992 Maintenance phase 4: *Implementation*
a. Input	•Original source code •Original project documentation •Detailed design from previous phase
b. Process	•Make code changes and additions •Perform unit tests •Review readiness for system testing
c. Control	•Inspect code •Verify CM control of new code Traceability of new code
d. Output	•Updated … …software …unit test reports …user documents
e. Selected quality factors	•Flexibility •Traceability •Comprehensibility •Maintainability •Reliability
f. Selected metrics	•Lines of code •Error rate

22. Maintenance

a) Activities and issues of software maintenance
b) Types of software maintenance
c) Maintenance techniques
d) Standardization
e) Management of maintenance
d) Metrics


A Typical Maintenance Flow
nominal path
Marketing
Written MR's
Customer
Maintenance manager
Help desk
Proposed M. R.'s
Approved M. R.'s
Maintenance engineer
Current source & documentation
Change control board
Modified source & documentation
Rejected MR's
Zoran Budimac, Univ. of Novi Sad. Parts adapted from Software Engineering: An Object-Oriented Perspective by E.Braude (Wiley 2001), with permission.
Graphics reproduced with permission from Corel.
35


Maintenance and patching
1. Interface with customer
Help desk
Complaints
Marketing
Docu-ment patch
Patch (optional)
Execute with patch
Zoran Budimac, Univ. of Novi Sad. Parts adapted from Software Engineering: An Object-Oriented Perspective by E.Braude (Wiley 2001), with permission.
36



Maintenance Patches

advantages

- Keeps customers satisfied in the short run
- Enables continued operation and testing without repeated prevalence of the defect
- Avoids masking other defects
- Enables test of fix

disadvantages

- Duplicates work
  - patch *and* final fix both implemented
- Sometimes never replaced
  - proper fix deferred forever!
- Complicates final fix
  - must remove
- Complicates documentation process

Ranked Problems in Maintenance (Deklava)

1. Changing priorities
2. Testing methods
3. Performance measurement
3. Incomplete or non-existent system documentation
5. Adapting to changing business requirements
6. Backlog size
7. Measurement of contributions
8. Low morale due to lack of recognition or respect
9. Lack of personnel, especially experienced
10. Lack of maintenance methodology, standards, procedures and tools . . . . .

Examples of Changing Priorities

Top priority . . .

. . . at release :

- ***Make this most bug-free game on the market***
  - **action: eliminate as many defects as possible**

. . . two months after release:

- ***Add more features than our leading competitor***
  - **action: add enhancements rapidly**

. . . six months after release:

- ***Reduce spiraling cost of maintenance***
  - **action: eliminate most severe defects only**

22. Maintenance

a) Activities and issues of software maintenance
b) Types of software maintenance
c) Maintenance techniques
d) Standardization
e) Management of maintenance
d) Metrics

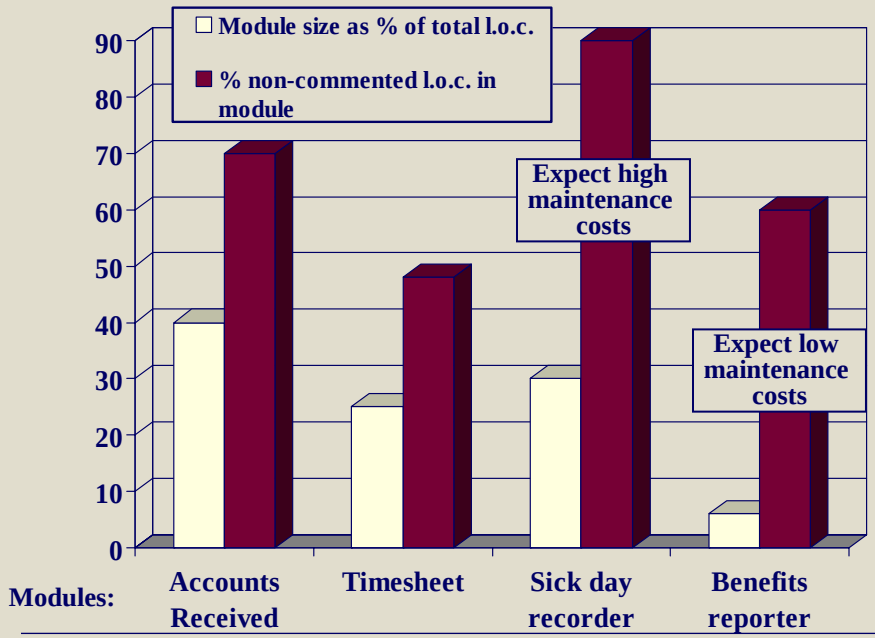
Maintenance Metrics

- **Number of lines of code under maintenance**
- **Person-months to perform various maintenance tasks**
- **Defect count**

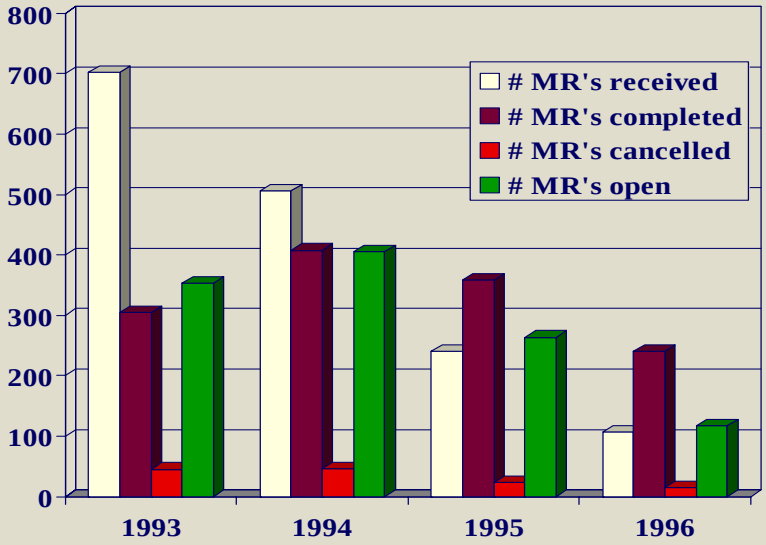
Maintenance Metrics Classified by Goal

Goal	Question	Selected Corresponding Metrics Note: The numbered metrics are from the IEEE.
Maximize customer satisfaction	*How many problems are affecting the customer?*	•(1) Fault density •(30) Mean time to failure •Break / fix ratio [Number of defects introduced by maintenance actions] / [Number of defects repaired]
	How long does it take to fix a problem?	•Fault closure Average time required to correct a defect, from start of correction work. •Fault open duration Average time from defect detection to validated correction.
	Where are the bottlenecks?	•Staff utilization per task type: Average person-months to (a) detect each defect and (b) repair each defect. •Computer utilization Average time / CPU time per defect.
Optimize effort and schedule	*Where are resources being used?*	Effort and time spent, per defect and per severity category … o… planning o… reproducing customer finding o… reporting error o… repairing o… enhancing
Minimize defects (continue focused development-type testing)	*Where are defects most likely to be found?*	•(13) Number of entries and exits per module •(16) Cyclotomic complexity

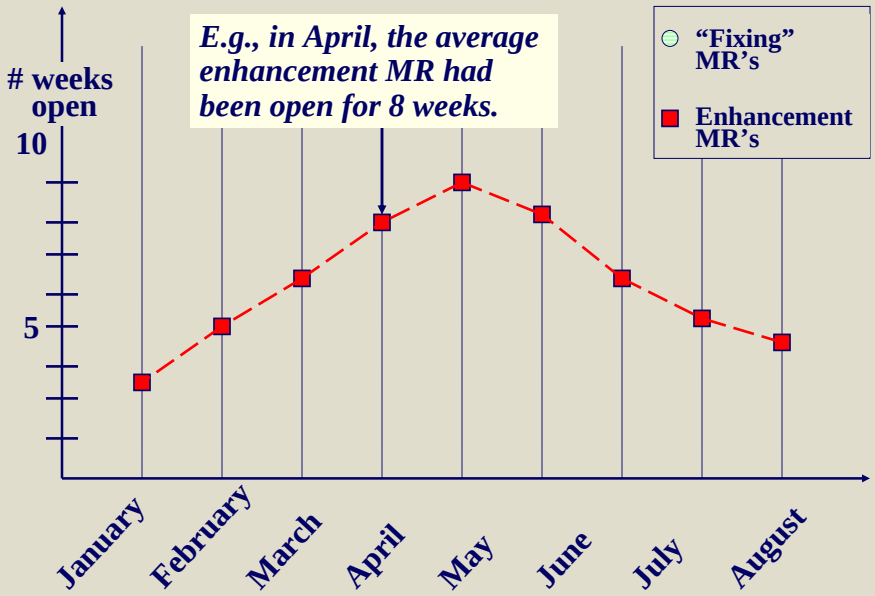
Predicting Relative Maintenance Effort



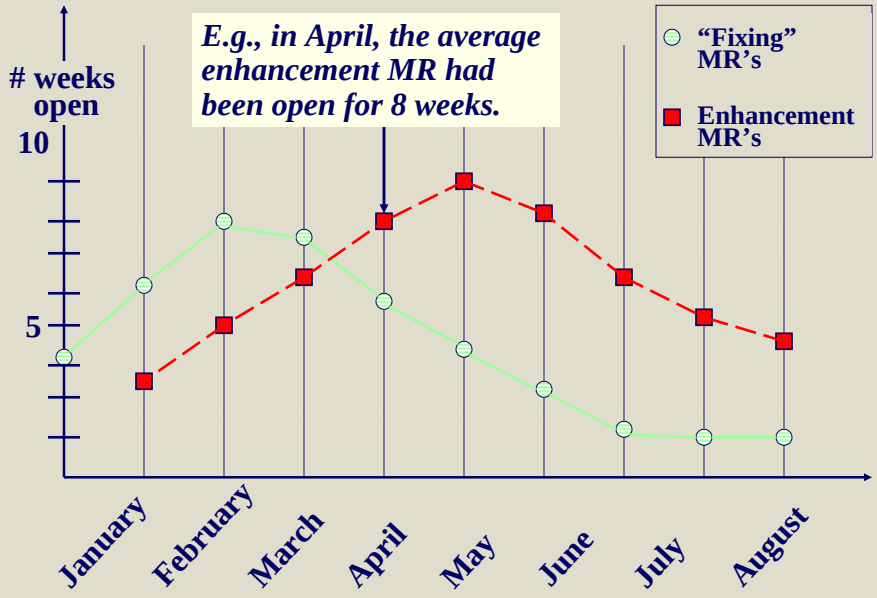
Managing Maintenance
Example profile of "fixing"-type MR's



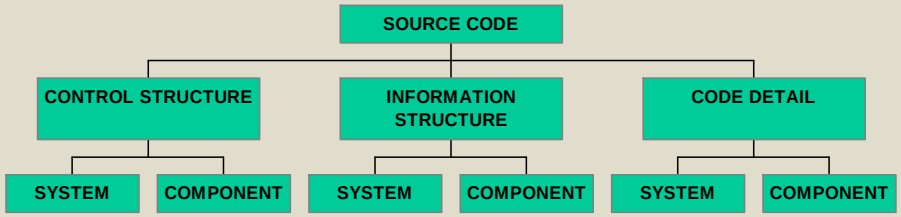
Profiles of Open Maintenance Requests



Profiles of Open Maintenance Requests



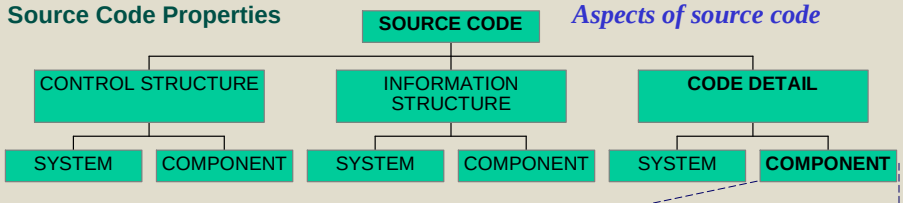
Effects on Maintainability of Source Code Properties



. . . .

Taken from Oman [Om1]

Effects on Maintainability of Source Code Properties



The maintainability of a product is affected by this property.

"+" means that *more* of this property usually makes an application *more maintainable;*

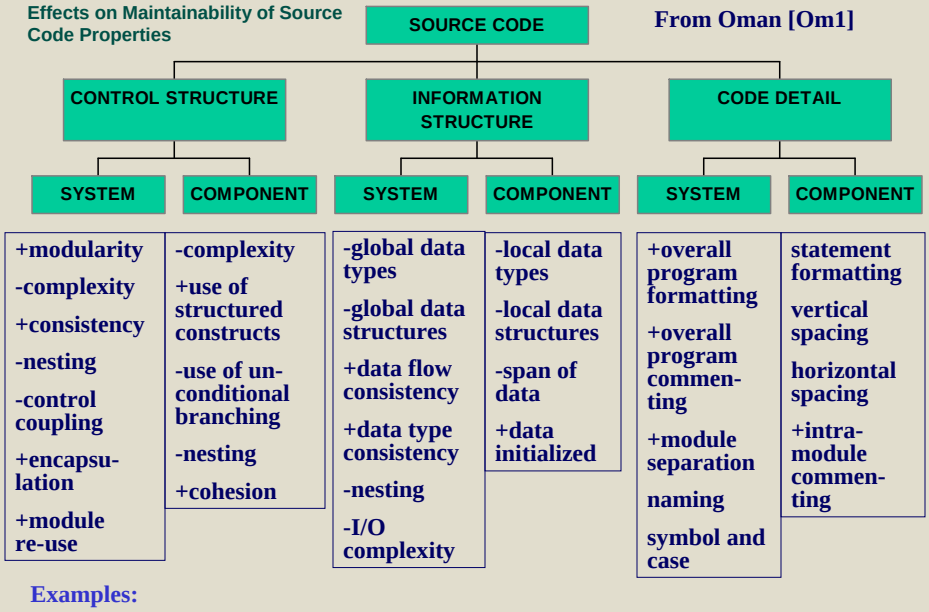
"-" means that more of the property usually makes an application *less maintainable.*

Taken from Oman [Om1]

- **statement formatting** *-- affects a product's maintainability, (but more is not necessarily better)*
- **vertical spacing**
- **horizontal spacing**
- **+ intra-module commenting** *-- usually, more comments with the code make a product more maintainable*

Effects on Maintainability of Source Code Properties

From Oman [Om1]



Examples:

+modularity* + means *greater modularity* usually makes an application *more maintainable;

-span of data* means that *the greater the scope of data structures*, the *less maintainable.

Examples - to do

- Examples for all four types from case study (here or at the end)