投票 最多可选1项

设置

Are you ready?

A Yes

B No

提交

Part 4
Project Management

Chapter 36

Delivering and Managing the System

Contents

36.1 Software Delivering

Before Delivering : Testing (Acceptance testing)

Helping users to understand and feel comfortable with the system

Training

Documentation

After delivering: Maintenance

36.1 Training

- **Users**: exercise the main system functions
- **Administrator** : perform supplementary functions
  - create back up copies of data files
  - define who has access to the system

- **User Training**
  - Introduces the primary functions
  - System transition: how the functions are performed now, how to perform later with the new system
- **Administrator Training**
  - How to bring up and run the new system
  - Support users

36.1 Documentation

Types of Documentations

User's manual

Operator's manual

General system guide

Tutorials and automated overviews

Other documentation: Programmer guide

User Helps and Troubleshooting

Contents

36.2 The Changing System

Software change is inevitable

- New requirements
- Business environment
- Bug fix
- Hardware and software update
- Software quality improvement

Key： Manage changes effectively!

36.2 The Changing System

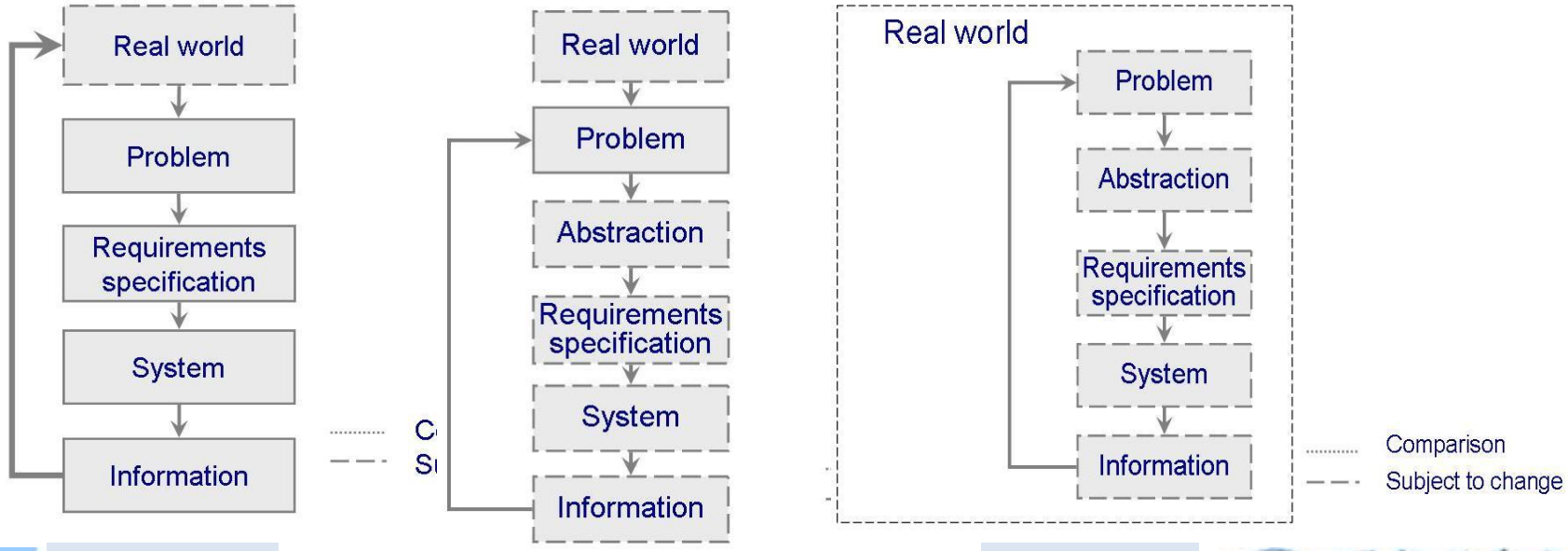
Lehman's System Types

S-system	formally defined, derivable from a specification; Matrix manipulation
P-system	requirements based on approximate solution to a problem, but real-world remains stable; Chess program
E-system	embedded in the real world and changes as the world does Software to predict how economy functions (but economy is not completely understood)

36.2 The Changing System

Lehman's System Types

S-system	**un-changed;**
P-system	**incremental change;**
E-system	**constant change;**



S-system

P-system

E-system

36.2 The Changing System

Software Evolution Policy

Software Maintenance

Software Reengineering

Contents

36.3 Software Maintenance

Maintenance:

Any work done to change the system after it is in operation.

IEEE: Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.

36.3 Software Maintenance

Types of Maintenance:

- **Corrective:** day-to-day functions
- **Adaptive:** system modifications
- **Perfective:** perfecting existing functions
- **Preventive:** performance from degrading to unacceptable levels

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

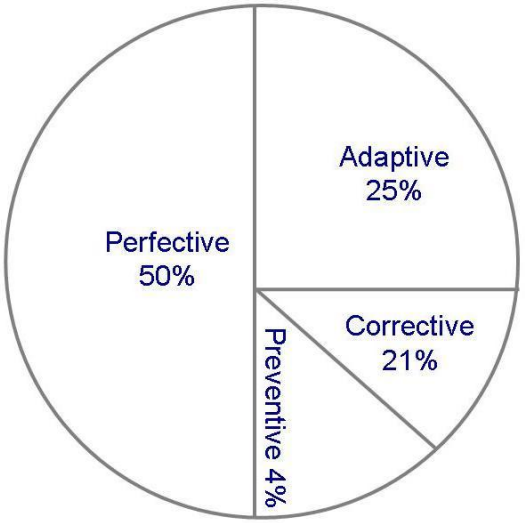
What is the proportion of **Perfective** maintenance among all maintenance?

A 25%

B 50%

C 60%

D 75%

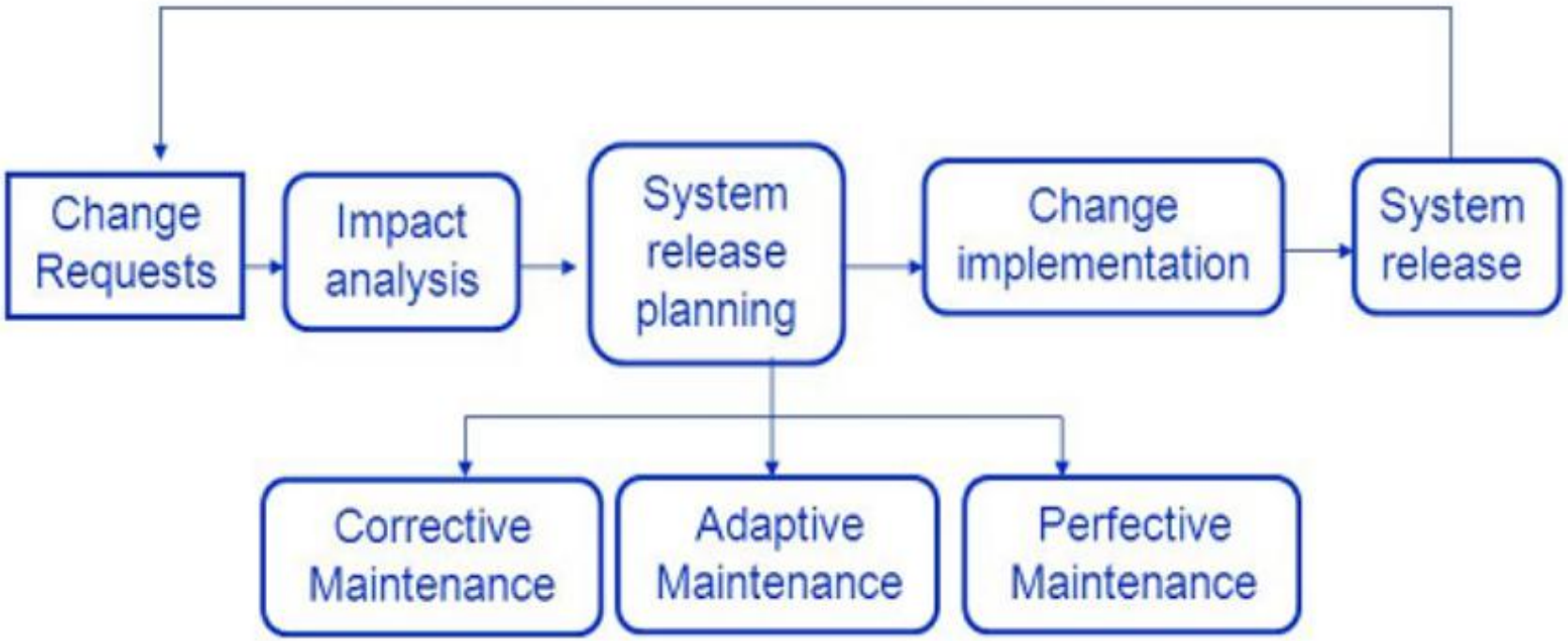


Effort (Lientz and Swanson)

提交

36.3 Software Maintenance

Process of Maintenance:

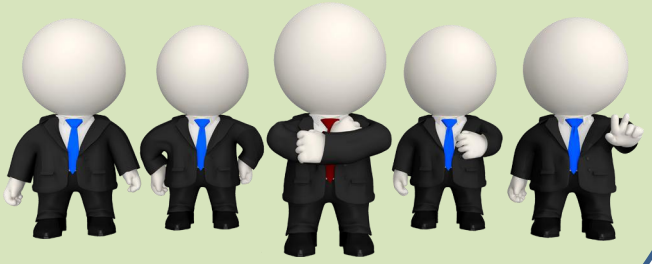


36.3 Software Maintenance

Who Performs Maintenance ?

Separate maintenance team

Part of development team



36.3 Maintenance Problem

Maintenance Problems:

- **Staff problems**
  - Limited understanding
  - Management priorities
- **Technical problems**
  - Artifacts and paradigms
  - Testing difficulties

Need to Compromise
(Depend on the type of maintenance)

36.3 Maintenance Problem

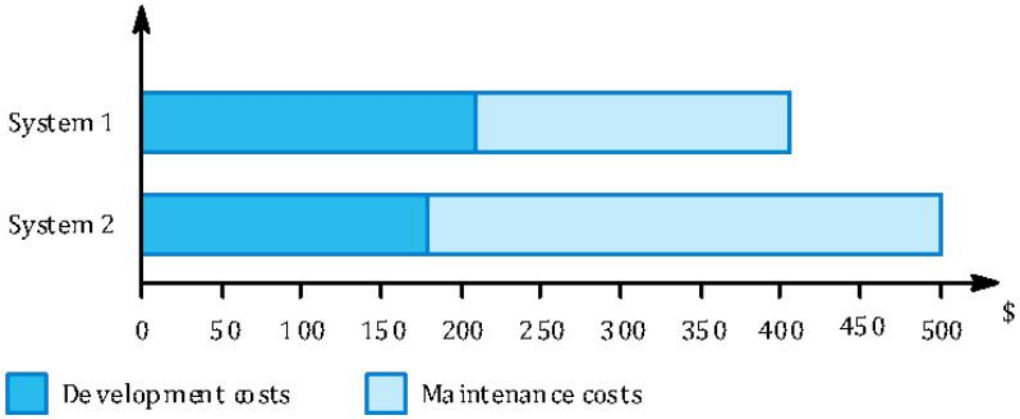
Software Maintenance Cost vs. Development Cost

Author	Maintenance as a % of Build Cost
Daniel D. Galorath	75%
Stephen R. Schach	67%
Thomas M. Pigoski	>80%
Robert L. Glass	40% – 80%
Jussi Koskinen	>90%

36.3 Software Maintenance

Maintenance Cost vs. Developent Cost:

- ✓ Business application software: 1:1
- ✓ Embedded realtime software: 4:1



Source: https://www.bilibili.com/video/BV1Q741157ve?p=83

36.3 Maintenance Problem

Factors Affecting Maintenance Effort

- **Application type**
- **Turnover and maintenance staff ability**
- **System life span (system age)**
- **Dependence on a changing environment**
- **Hardware characteristics**
- **System quality**
  - Design quality
  - Code quality
  - Documentation quality
  - Testing quality

36.3 Maintenance Problem

Modeling Maintenance Effort: Belady and Lehman (1972)

$$M = p + K^{c-d}$$

M : total maintenance effort
p : productive effort, including analysis, design, code, testing
c: complexity caused by lack of structured design and document
d : degree of maintenance team familiarity with the software
K : empirical constant, depends on the environment

36.3 Maintenance Problem

Modeling Maintenance Effort: Sample

$$M = p + K^{c-d}$$

The development effort for a software project is 500 person months. The empirically determined constant (K) is 0.3. The complexity of the code is quite high and is equal to 8. Calculate the total effort expended (M) if

(i) maintenance team has good level of understanding of the project (d=0.9)

(ii) maintenance team has poor understanding of the project (d=0.1)

36.3 Maintenance Problem

Modeling Maintenance Effort: Sample

$$M = p + K^{c-d}$$

Solution

Development effort (P) = 500 PM

K = 0.3

C = 8

(i) maintenance team has good level of understanding of the project (d=0.9)

$$M = P + Ke^{(c-d)}$$
$$= 500 + 0.3e^{(8-0.9)}$$
$$= 500 + 363.59 = 863.59 \text{ PM}$$

(ii) maintenance team has poor understanding of the project (d=0.1)

$$M = P + Ke^{(c-d)}$$
$$= 500 + 0.3e^{(8-0.1)}$$
$$= 500 + 809.18 = 1309.18 \text{ PM}$$

36.3 Maintenance Problem

Modeling Maintenance Effort: COCOMO II

$$\text{Size} = ASLOC\,(AA + SU + 0.4DM + 0.3CM + 0.3IM)/100$$

ASLOC: number of source lines of code to be adapted
AA: assessment and assimilation effort
SU: amount of software understanding required
DM: percentage of design to be modified
CM: percentage of code to be modified
IM: percentage of external code to be integrated

36.3 Measuring Maintenance Characteristics

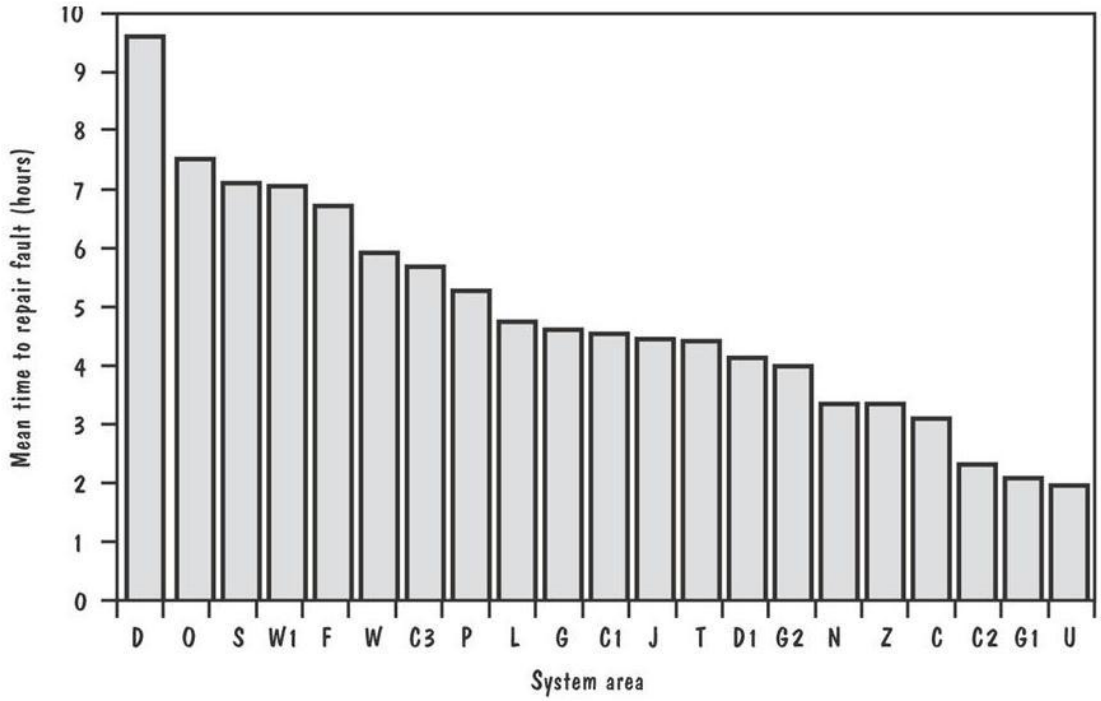
Maintenance Measurement - External view

- Necessary measures
  - time at which problem is reported
  - time lost due to administrative delay
  - time required to analyze problem
  - time required to specify which changes are to be made
  - time needed to make/test/document the change
- Desirable measures
  - ratio of total change implementation time to total number of changes implemented
  - number of unresolved problems
  - time spent on unresolved problems
  - percentage of changes that introduce new faults
  - number of components modified to implement a change

36.3 Measuring Maintenance Characteristics

Maintenance Measurement

- External view (Mean time to repair)



the various subsystems for software at a large British firm

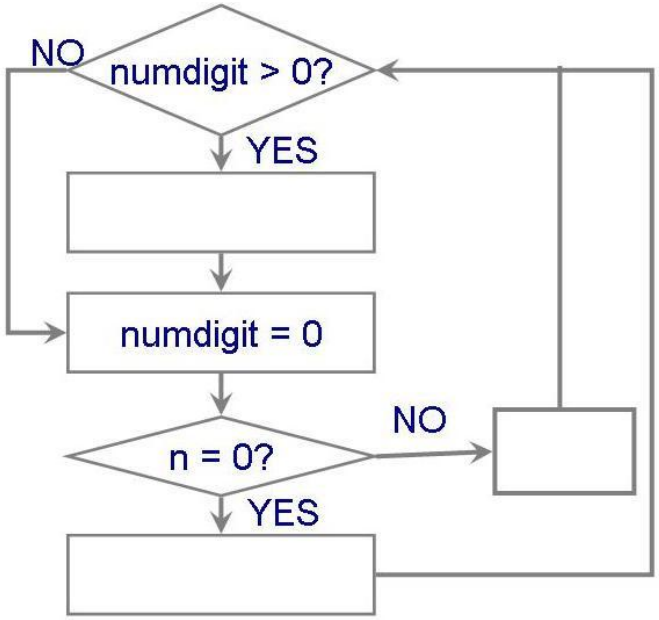
36.3 Measuring Maintenance Characteristics

Maintenance Measurement - Internal view

- Complexity (Cyclomatic number , McCabe)

The structural complexity of the source code

(linearly independent **path = e - n + 2, e: edges, n : nodes**)



CONTROL FLOW GRAPH

EQUIVALENT GRAPH

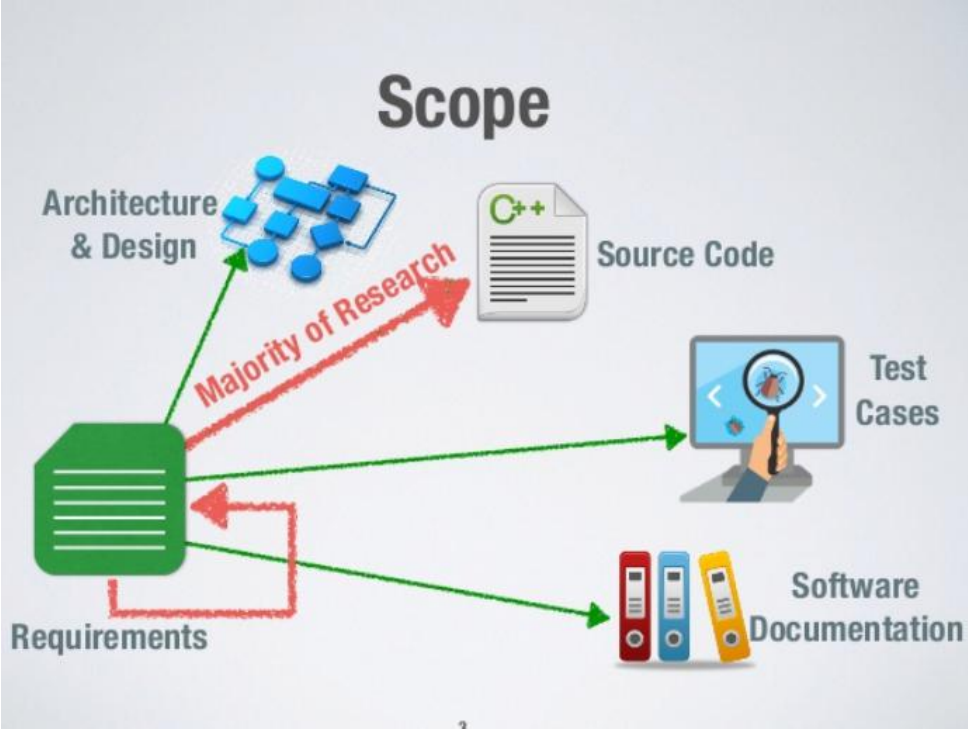
e =8, n=6
=> path =8-6+2
=4

Another:
path = decision + 1
= 3 + 1
= 4
(3 = 2 while + 1 if)

36.3 Measuring Techniques and Tools

Change Impact analysis：

Identifies potential effects of proposed software changes



36.3 Measuring Techniques and Tools

Measuring Change Impact analysis

1. Workproduct:

development artifacts whose change is significant

2. Horizontal traceability:

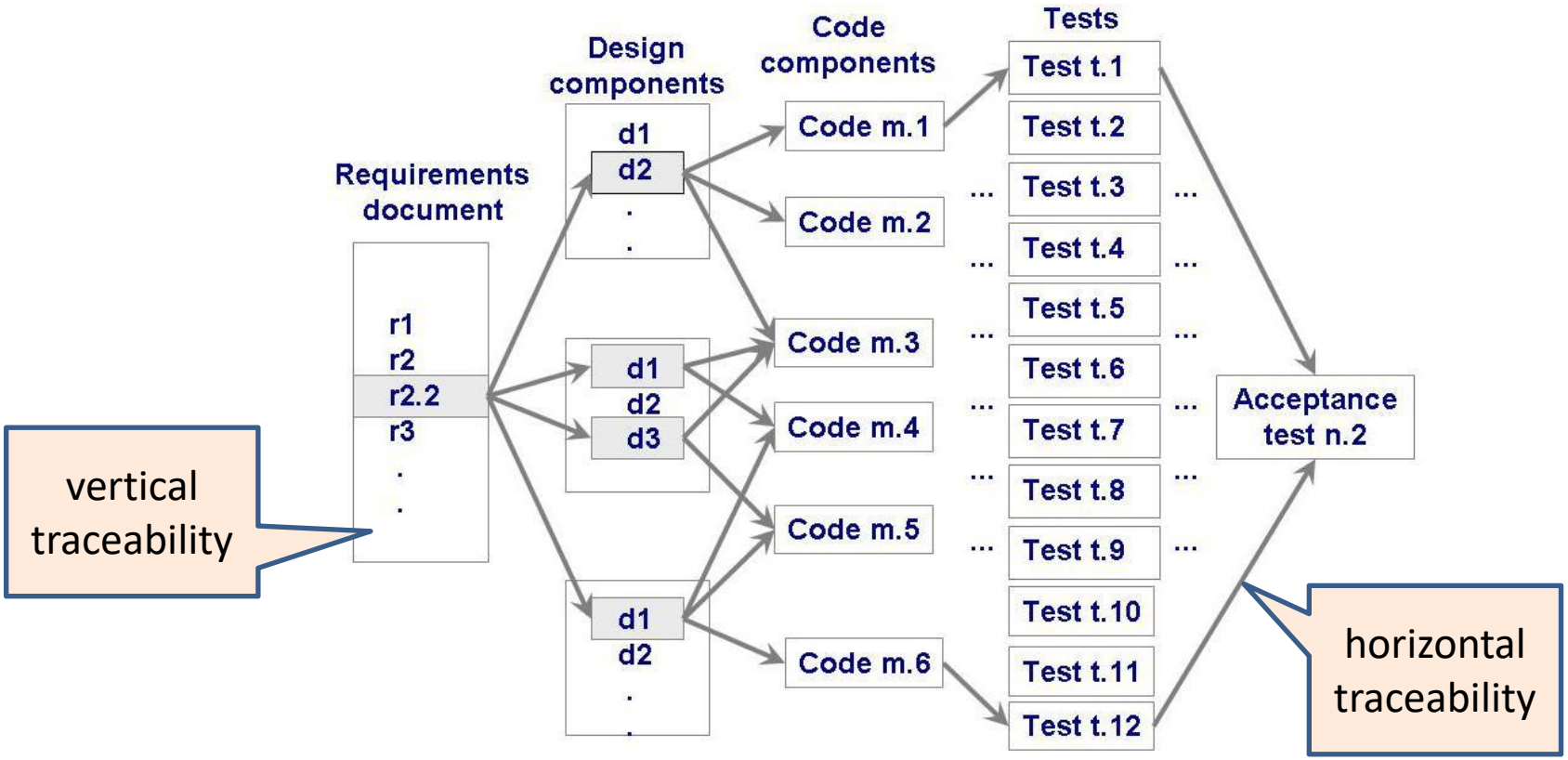
relationships of components across collections of workproducts

3. Vertical traceability:

relationships among parts of a workproduct

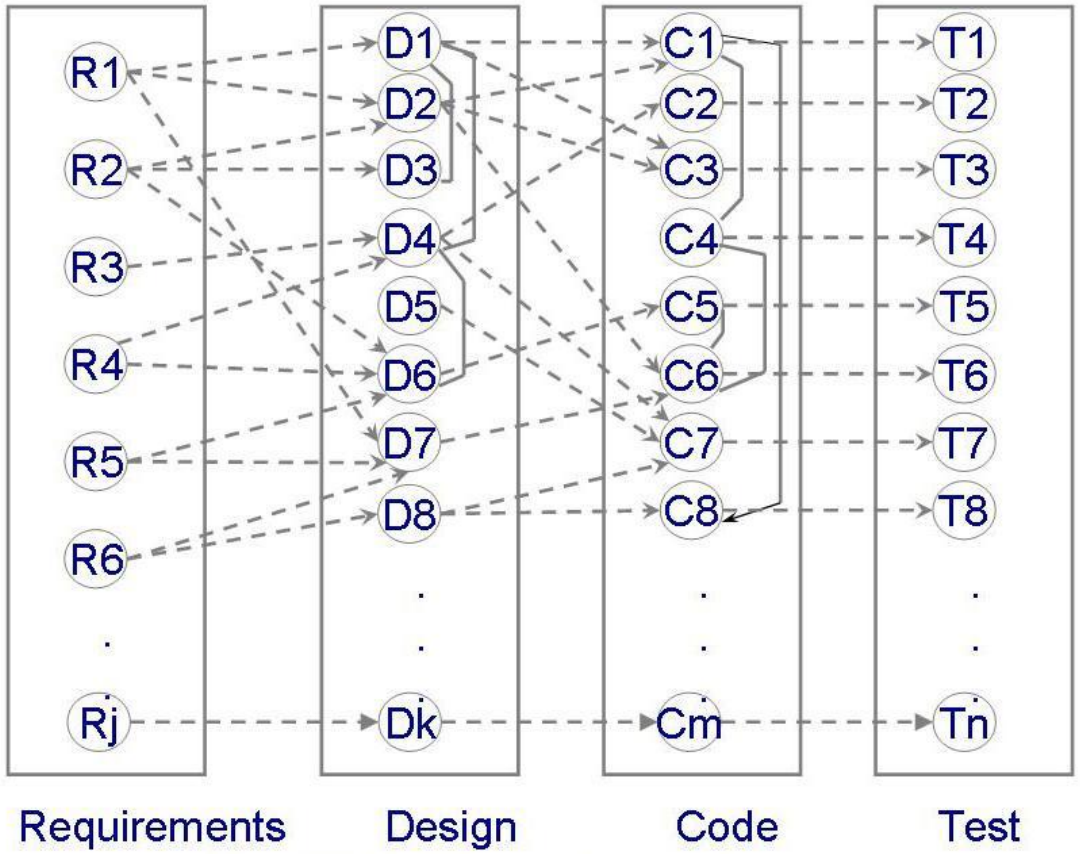
36.3 Measuring Techniques and Tools

Traceability: traceability links among/across related workproducts



36.3 Measuring Techniques and Tools

Underlying graph



Measures:

- the number of nodes
- in-degree
- out-degree
- Complexity
-...

Method:

Compare before and after change

Decision:

1. change or not
2. implement way

36.3 Measuring Techniques and Tools

Tools

Configuration management

Text editors

File comparators

Compilers and linkers

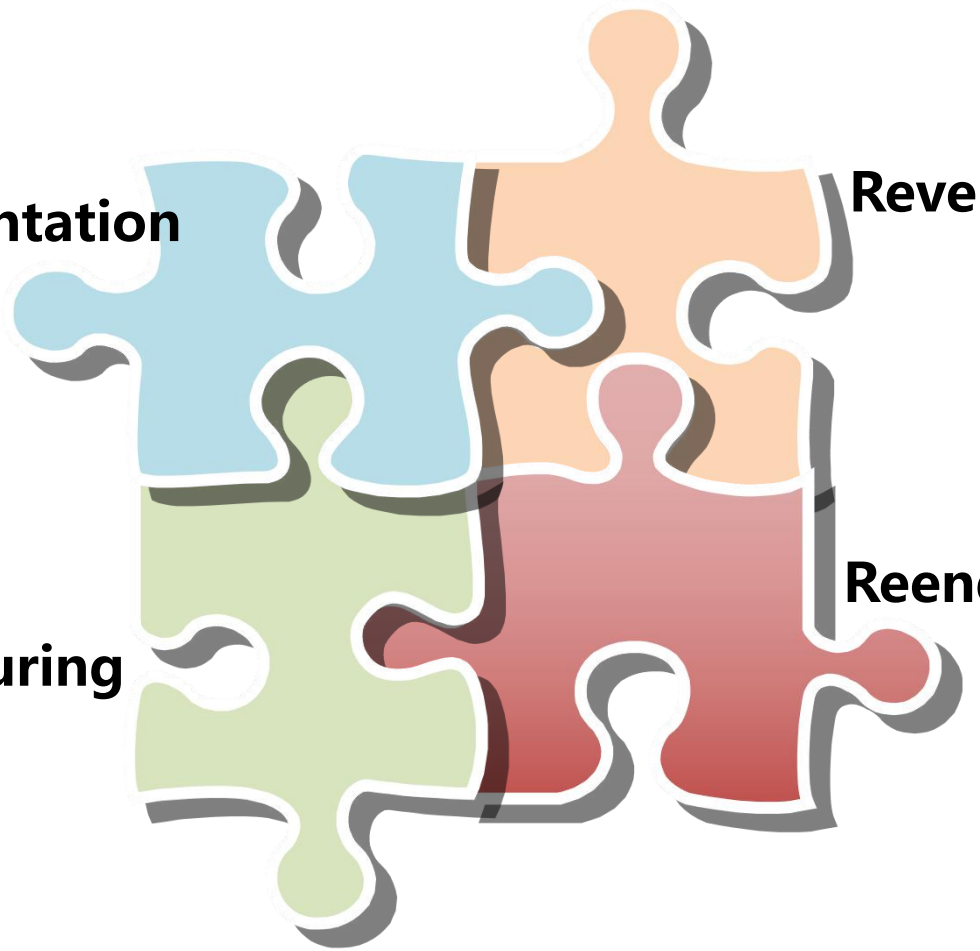
Debugging tools

Cross-reference generators

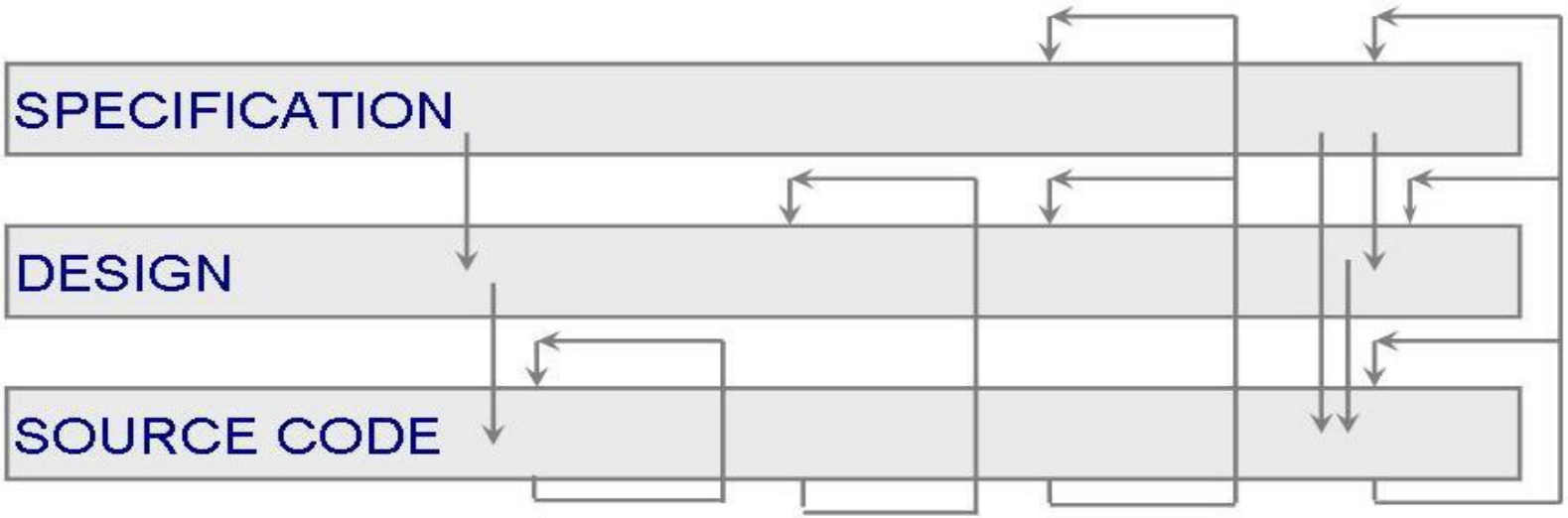
Static code analyzers

Contents

36.4 Software Reengineering



36.4 Software Reengineering



Forward Engineering
- forward progression through process

Restructuring
- from code
- internally represent
- Iteratively simplify structure and eliminate dead code
- Regenerate code

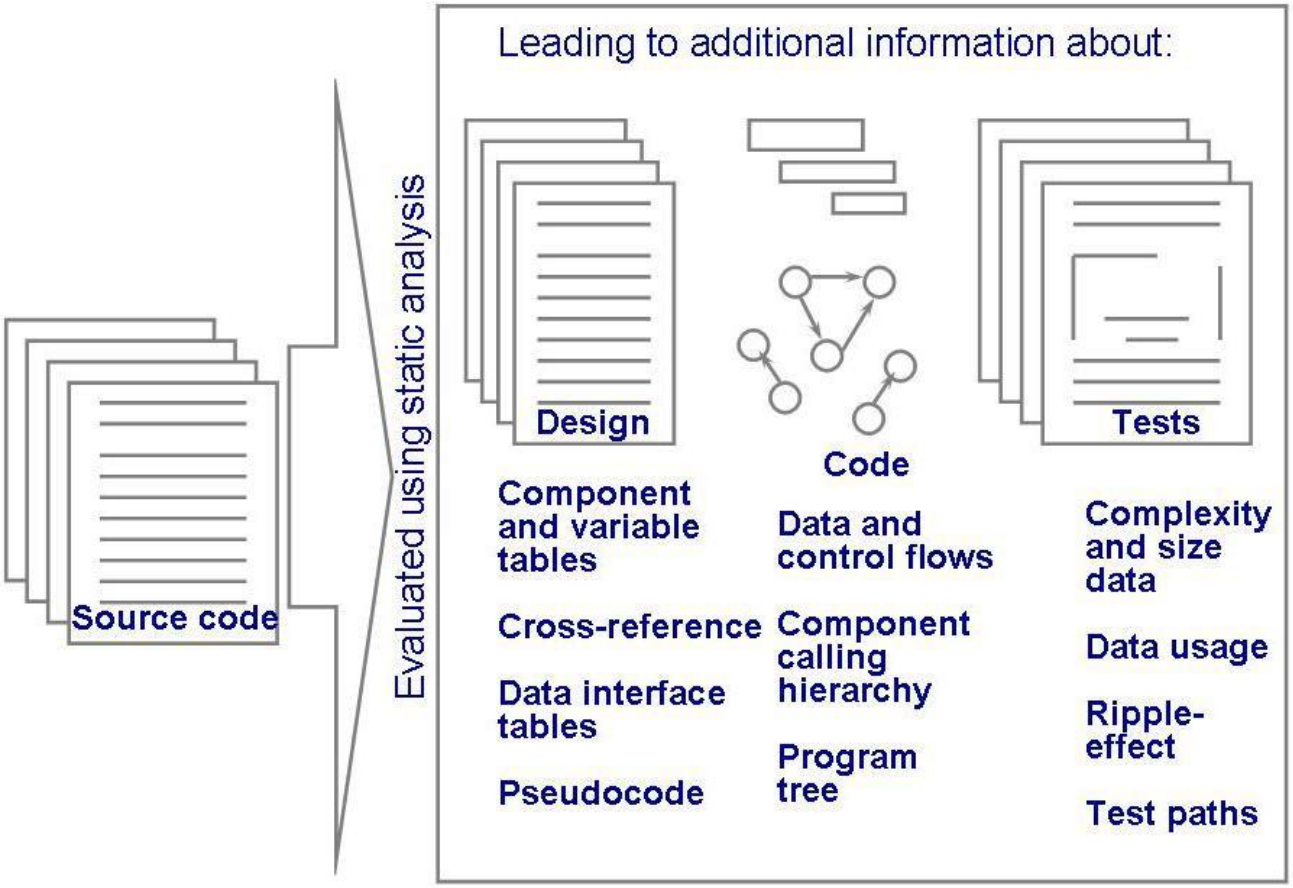
Redocumenting
- from code
- static analysis reports on structure, complexity, volume, data, etc.
- not based on software methods

Reverse Engineering
- from code
- produces design and specification based on accepted software methods
- manages representation

Reengineering
- from code
- reverse-engineer code
- forward-engineer: complete and modify representation
- regenerate code

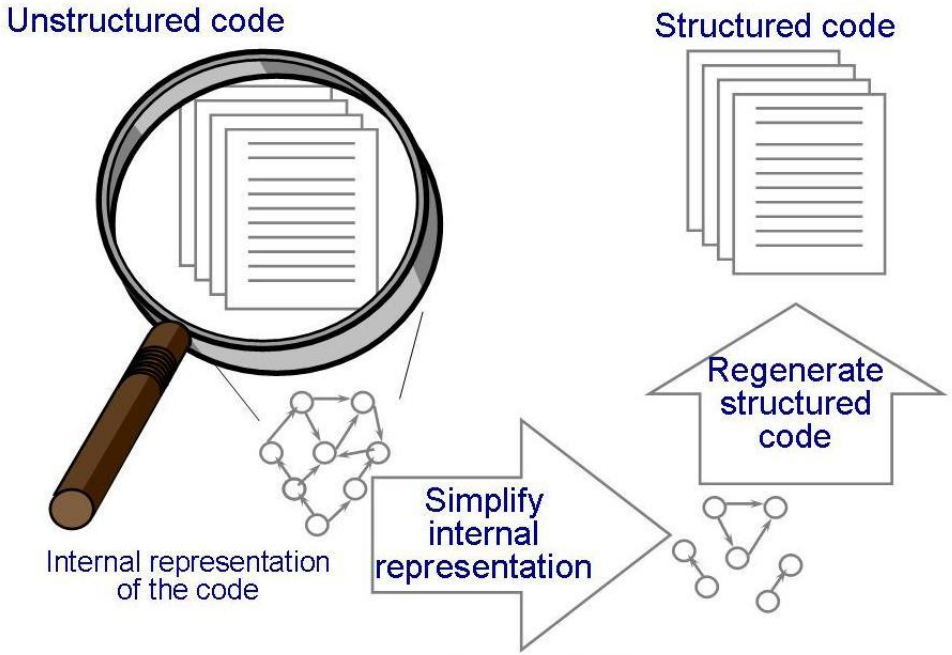
36.4 Software Reengineering

Redocumentation process



36.4 Software Reengineering

Restructuring process



three major activities involved in restructuring:

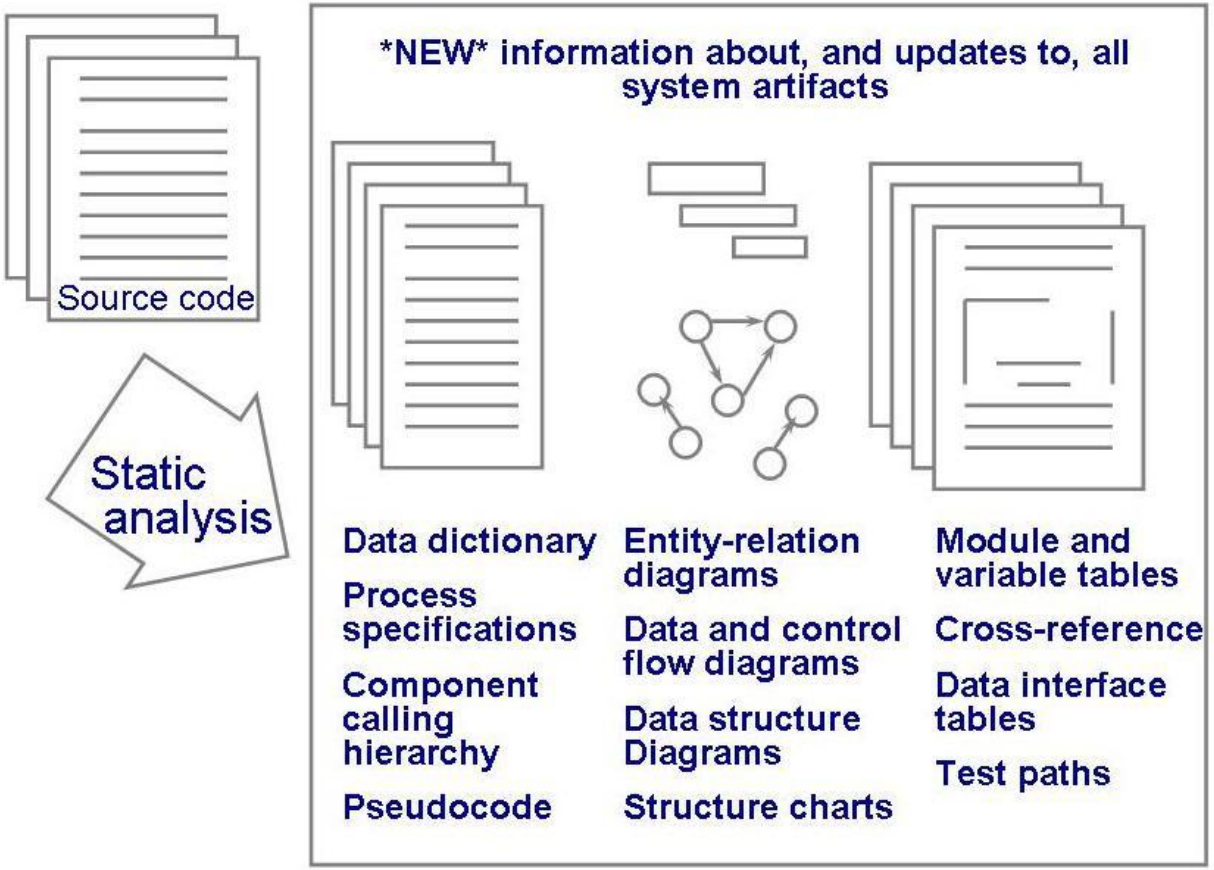
(1) static analysis

(2) simplification of the representations

(3) refined representation used to generate a structured version

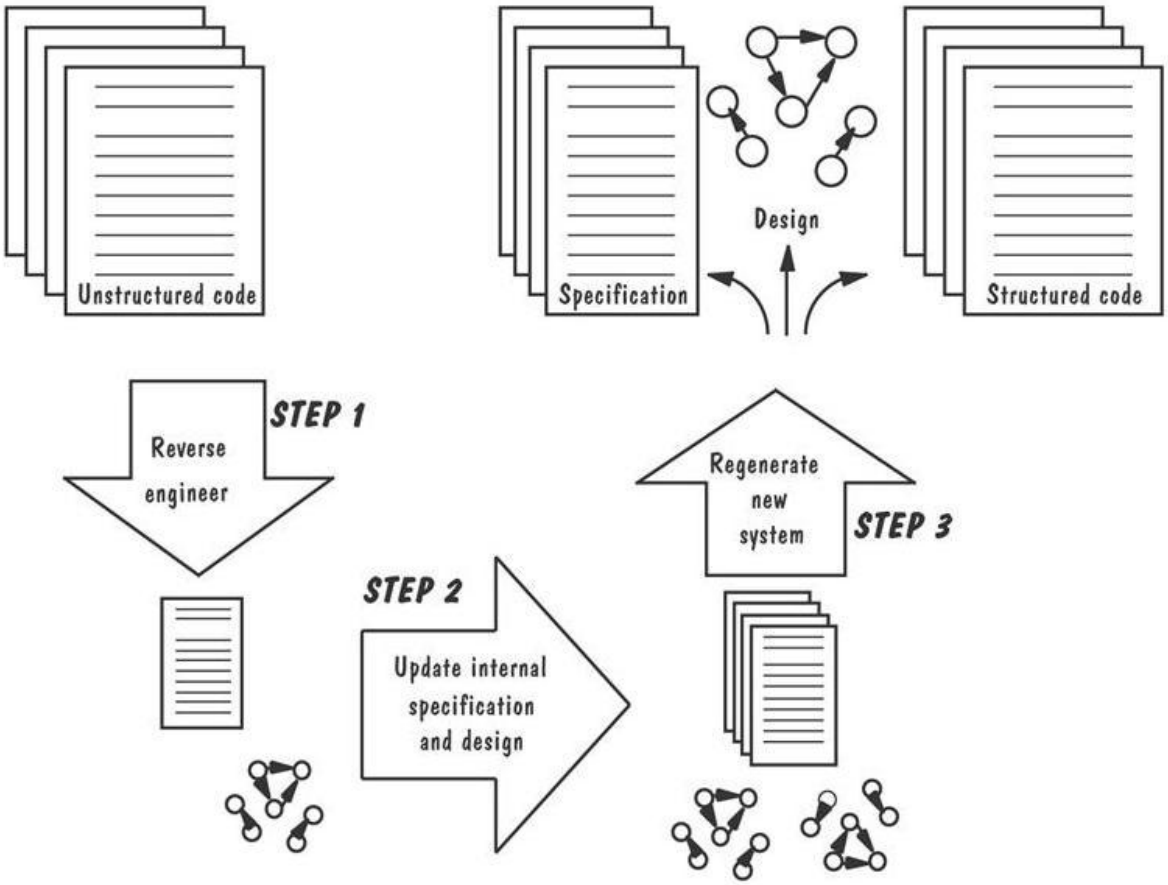
36.4 Software Reengineering

Reverse Engineering process



36.4 Software Reengineering

Reengineering process



Summary

1. Delivering : training and documentation
2. Software Evolution Policy: maintenance , reengineering
3. Maintenance type: corrective, adaptive, perfective, preventive
4. Maintenance effort estimation: Belady and Lehman, COCOMO II
5. Maintenance metrics: complexity, mean time to repair, number/ratio of changes
6. Maintenance techniques: change impact analysis
7. Reengineering: Redocumentation, Reverse engineering, Restructuring, Reengineering

The End