Software Maintenance (II)

Software Maintenance: Key challenges

(poor) quality of documentation

User demand for enhancements and extensions

Competing demands for maintainers' time

Difficulty in meeting scheduled commitments

Turnover in user organizations

Software Maintenance: Key challenges

Limited understanding

47% of software maintenance effort devoted to understanding the software

System has “m” components, we need to change “k” of them. Thus there are k*(m-k) + k*(k-1)/2 interfaces to check for impact and correctness

50% of effort can be attributed to lack of user understanding (i.e., incomplete or mistaken reports of errors and enhancements)

Low morale

Software maintenance is regarded as less interesting than development

Main factors (1)

Application type
Systems with timing issues (real-time and highly synchronized);
Systems with rigidly defined data formats…

System novelty

Turnover and maintenance staff availability

System life span

Dependence on the changing environment
(P, E systems…)

Hardware characteristics

Main factors (2)

Design quality
Independent, cohesive components, well-defined architecture

Code quality

Documentation quality

Testing quality

Maintenance management

- Maintenance has a poor image amongst development staff as it is not seen as challenging and creative
- Maintenance costs increase as the software is maintained
- The amount of software which has to be maintained increases with time
- Inadequate configuration management often means that the different representations of a system are out of step

Maintenance cost factors

- Module independence
  - It should be possible to change one module without affecting others
- Programming language
  - High-level language programs are easier to maintain
- Programming style
  - Well-structured programs are easier to maintain
- Program validation and testing
  - Well-validated programs tend to require fewer changes due to corrective maintenance

Maintenance cost factors

- Documentation
  - Good documentation makes programs easier to understand
- Configuration management
  - Good CM means that links between programs and their documentation are maintained
- Application domain
  - Maintenance is easier in mature and well-understood application domains
- Staff stability
  - Maintenance costs are reduced if the same staff are involved with them for some time

Maintenance cost factors

- Program age
  - The older the program, the more expensive it is to maintain (usually)
- External environment
  - If a program is dependent on its external environment, it may have to be changed to reflect environmental changes
- Hardware stability
  - Programs designed for stable hardware will not require to change as the hardware changes

Maintenance metrics

Control complexity

Can be measured by examining the conditional statements in the program

Data complexity

Complexity of data structures and component interfaces.

Length of identifier names

Longer names imply readability

Program comments

more comments mean easier maintenance

Maintenance metrics

Coupling

How much use is made of other components or data structures

Degree of user interaction

The more user I/O, the more likely the component is to require change

Speed and space requirements

Require tricky programming, harder to maintain

Process metrics

- Number of requests for corrective maintenance
- Average time required for impact analysis
- Average time taken to implement a change request
- Number of outstanding change requests
- If any or all of these is increasing, this may indicate a decline in maintainability

Halstead metrics (software physics)

Syntax elements of a program:
Number of unique operators (n1)
Number of unique operands (n2)
Total occurrence of operators (N1)
Total occurrence of operands (N2)

Length	N = N1+ N2
Vocabulary	n = n1+n2
Volume	V = N(log2(n))
Difficulty	D = (n1/N1) * (n2/N2)
Effort	E = D*V
Time	T=E/18
Faults	$E^{2/3}/3000$

radon 5.1.0

```
pip install radon
```

- **cc**: compute Cyclomatic Complexity
- **raw**: compute raw metrics
- **mi**: compute Maintainability Index
- **hal**: compute Halstead complexity metrics

CC score	Rank	Risk
1 - 5	A	low - simple block
6 - 10	B	low - well structured and stable block
11 - 20	C	moderate - slightly complex block
21 - 30	D	more than moderate - more complex block
31 - 40	E	high - complex block, alarming
41+	F	very high - error-prone, unstable block

Block type	Letter
Function	F
Method	M
Class	C

```
(venv) (base) Witolds-iMac-Pro:pythonProject8 witold$ radon --h
usage: radon [-h] [-v] {cc,raw,mi,hal} ...

positional arguments:
  {cc,raw,mi,hal}
    cc              Analyze the given Python modules and compute Cyclomatic Complexity (CC).
    raw             Analyze the given Python modules and compute raw metrics.
    mi              Analyze the given Python modules and compute the Maintainability Index.
    hal             Analyze the given Python modules and compute their Halstead metrics.

optional arguments:
  -h, --help        show this help message and exit
  -v, --version     show program's version number and exit
(venv) (base) Witolds-iMac-Pro:pythonProject8 witold$
```

4: Run | TODO | 6: Problems | Terminal | Python Console

Maintainability index (1)

$$Maintainability = 171 - 5.2 * \ln(V) - 0.23 * CC - 16.2 * \ln(LOC) + 50 * \mathrm{sqrt}(2.46 * perCOM)$$

- Volume - V
- Cyclomatic Complexity - CC
- Lines of Code - LOC
- % of Comments - perCOM

Within HP: engineers asked to rate the maintainability for 16 projects on a 0 to 100 (excellent –fully maintainable) scale.

Maintainability index (2)

$$\text{Maintainability} = 171 - 5.2 * \ln(V) - 0.23 * CC - 16.2 * \ln(LOC) + 50 * \text{sqrt}(2.46 * \text{perCOM})$$

- => 85 - Highly Maintainable
- 65 - 85 - Moderately Maintainable
- <= 65 - Difficult to Maintain

Maintainability index (3)

- Visual Studio (Microsoft); updated formula 2011

Maintainability = Max(0,(171 - 5.2 * ln(HV) - 0.23 * (CC) - 16.2 * ln(LOC))*100 / 171)

HV-Halstead volume

- => 20 - Highly Maintainable
- => 10 & < 20 - Moderately Maintainable
- <10 - Difficult to Maintain

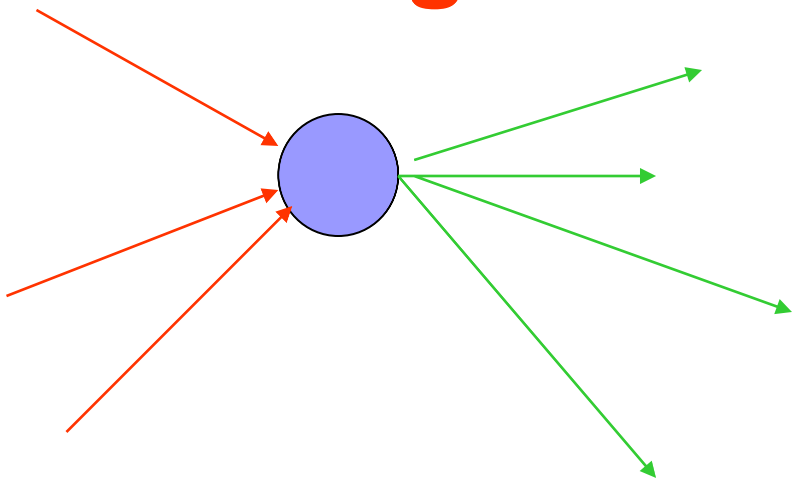
Measuring maintenance characteristics

Note: maintainability is not restricted to code – could apply to different software products (specification, design, tests, documentation)

External view of maintainability
expressed as <u>mean time to repair</u>
(we should know the time at which the problem is reported, time required to analyze the problem, to specify which change are to be made, time needed to make the change, test the change, document the change)

Internal view of maintainability
Complexity of code,
code size,
fan-in and fan-out characteristics → [fan-in * fan_out]^2
quality of documentation…

Traceability graph- evaluation of risk of change



in-degree [the number of edges the node is a destination of]
out-degree **[**the number of edges the node is a source of]
Complexity measure **of node evaluated before and after the change**
(e.g., cyclomatic complexity)

Risk = f(in-degree, out-degree, complexity measure of node)

Traceability

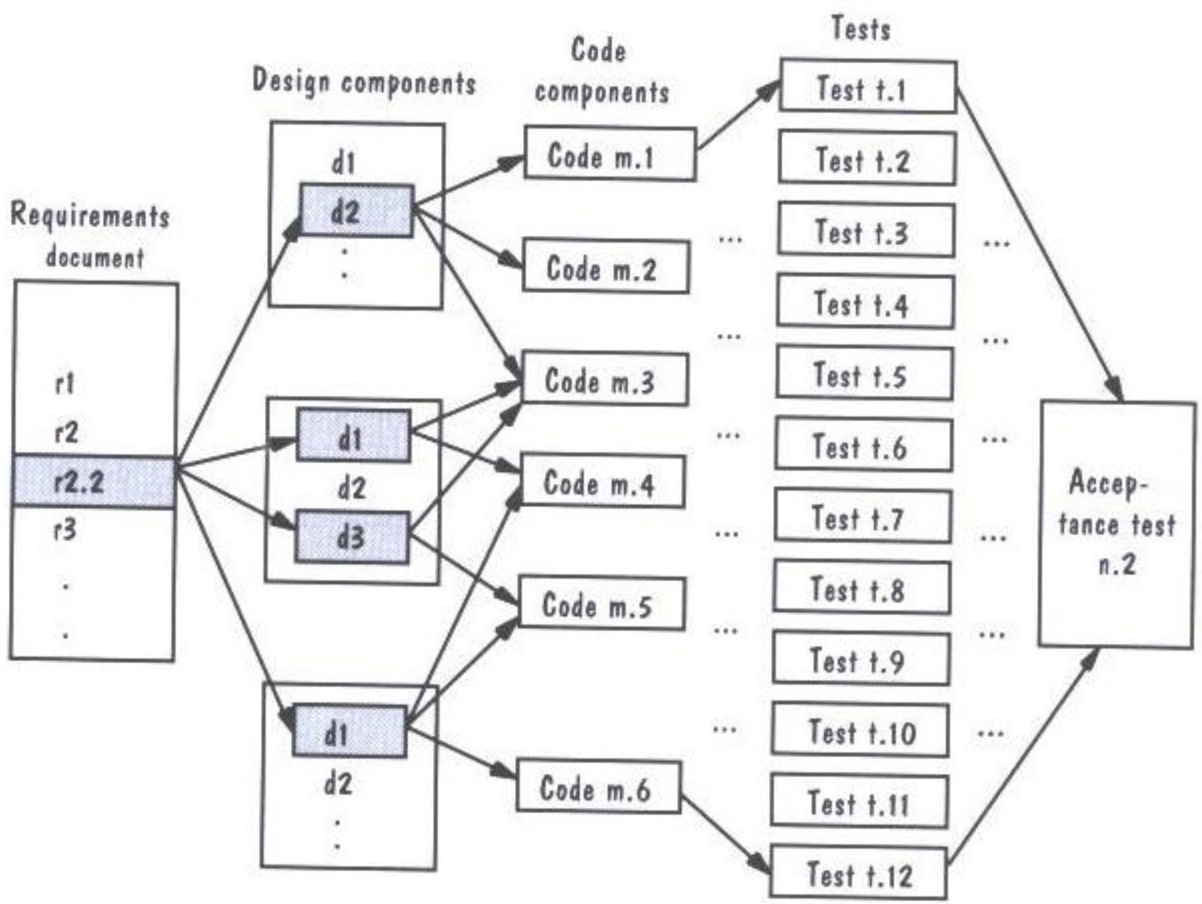
workproduct: any development artifact whose undergoes change
(say, requirements, design, code components, test cases, documentation)

IMPACT OF CHANGE for all workproducts

Vertical traceability : expresses the relationships among the parts of the workproduct (e.g., interdependencies among system requirements); <u>product view of change</u>

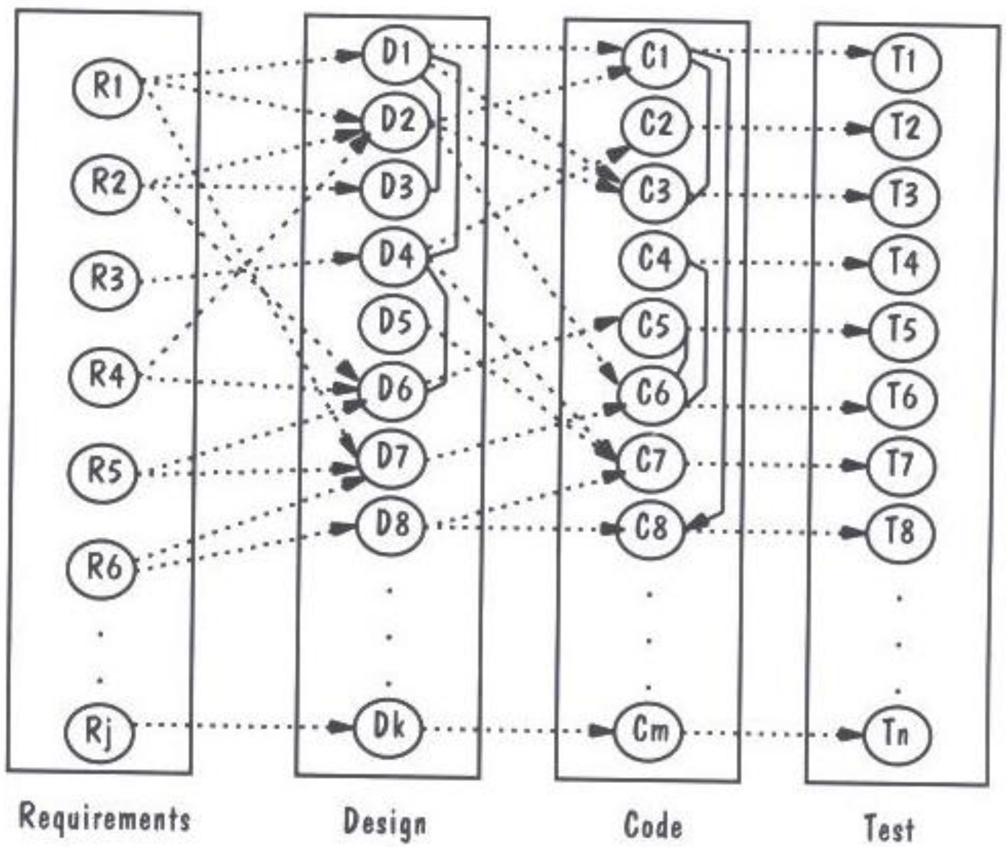
Horizontal traceability: expresses the relationships of the components across the collections of workproducts; <u>process view of change</u>

Horizontal traceability: example



process view of change

Traceability graph



Vertical traceability ————————

Horizontal traceability:..........................

product view of change

Main causes of unmaintainable software

National Institute of Science & Technology (NIST)

- Poor software design
- Poorly coded software
- Software designed for outdated hardware
- Lack of common data definitions
- Use of multiple languages in one program
- Grown software inventory
- Excessive resource requirements
- Inadequate documentation
- Inadequate user interface
- Lack of highly skilled staff