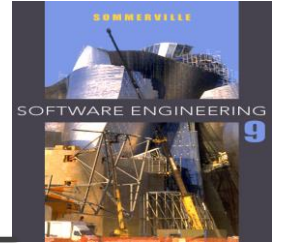


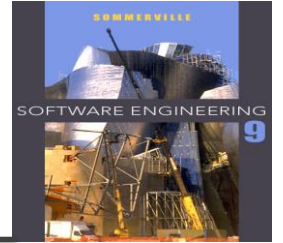
Chapter 5 – Software Testing and Maintenance

Topics Covered



- ✧ Software verification and validation
- ✧ Software Testing
- ✧ Stages of Testing
- ✧ Black Box Testing (BBT)
- ✧ White Box Testing (WBT)
- ✧ Software Maintenance and its types
- ✧ The Software Maintenance Process
- ✧ Maintenance Costs and Prediction
- ✧ Software RE-Engineering

5.1 Software Verification and Validation



- ✧ Verification and validation are the processes in which we check a product against its specifications and the expectations of the users who will be using it.

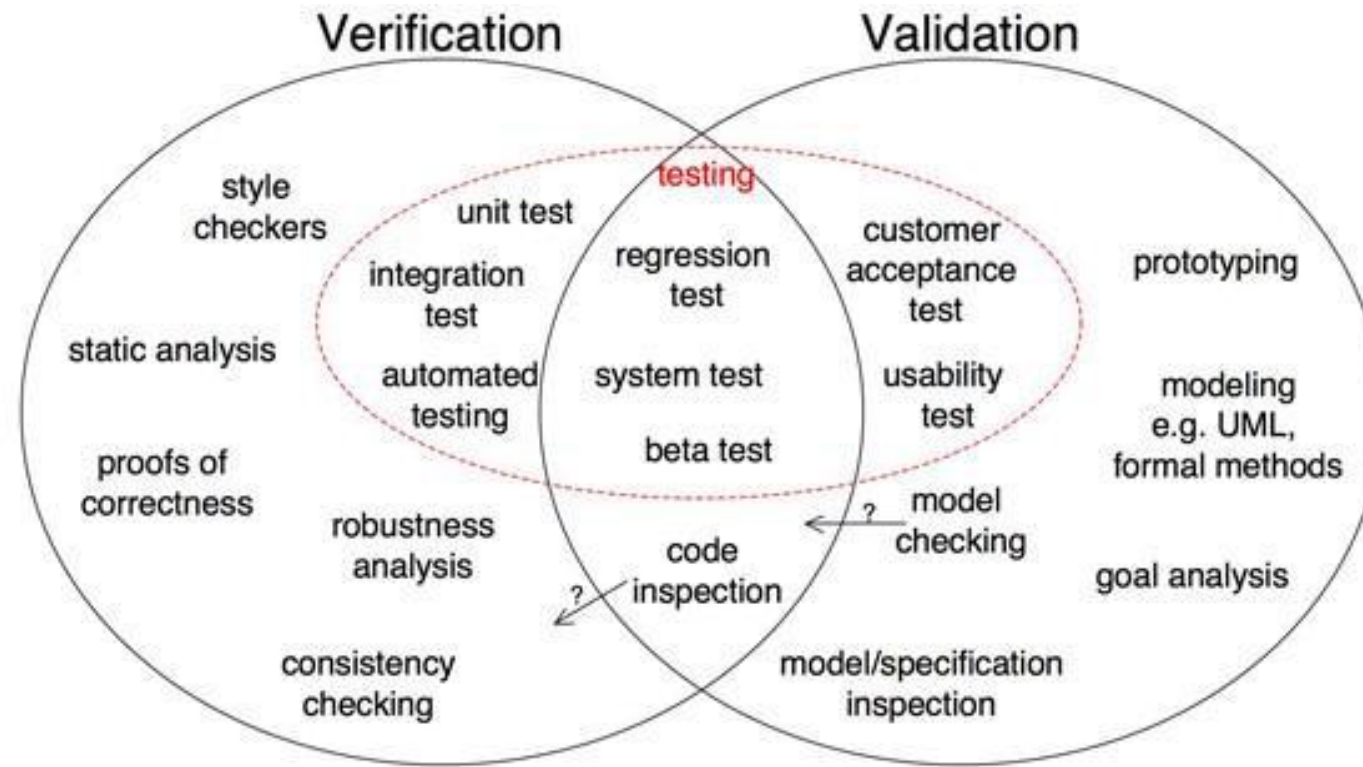
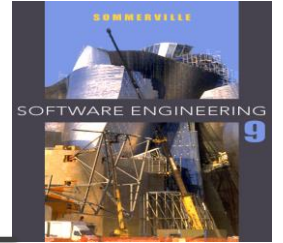


Figure 7.1: Scope of V & V in software engineering

Verification vs Validation



✧ Verification:

"Are we building the product right"

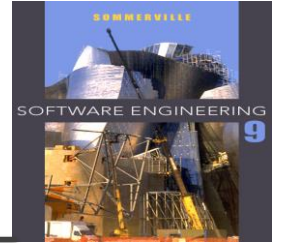
"Does the product meet system specifications?"

✧ Validation:

"Are we building the right product"

"Does the product meet user expectations?"

5.2 Software Testing



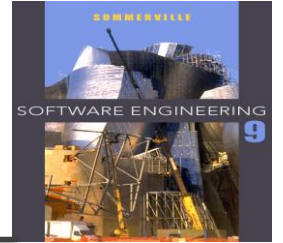
✧ **Testing** is the process of executing a program with the intent of finding errors.

“Testing can reveal the presence of errors NOT their absence”

5.2.1 Why should we Test?

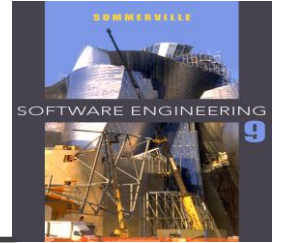
- ✧ Software testing is an expensive activity but launching of software without testing may lead to cost much higher than the cost of testing itself.
- ✧ Testing assure to developers and customers that the software meets its requirements.
 - For custom software (be-spoke products) there should be at least one test case for every requirement.
 - For generic software products there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.
- ✧ It discovers the situations where behavior of the software is incorrect, undesirable or does not match to its specification.
- ✧ Testing identifies undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption.

5.2.2 Who should do the Testing?



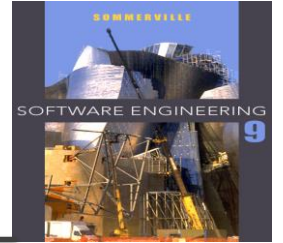
- ✧ Testing requires the developers to find errors from their software.
- ✧ It is difficult for software developer to point out errors from own creations.
- ✧ Many organizations have made a distinction between development and testing phase by making different people responsible for each phase.

5.2.3 What should we Test?



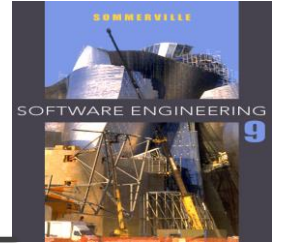
- ✧ We should test the program's responses to every possible input. It means, we should test for all valid and invalid inputs.
 - Suppose a program requires two 8 bit integers as inputs. Total possible combinations are 28×28 . If only one second it required to execute one set of inputs, it may take 18 hours to test all combinations. Practically, inputs are more than two and size is also more than 8 bits. Moreover invalid inputs have to be tested as well so complete testing is not possible.
- ✧ A strategy should develop test cases for the testing of small portion of program and also test cases for complete system or function.


5.3 Stages of Testing



- ✧ **Development testing**, where the system is tested during development to discover bugs and defects.
- ✧ **Release testing**, where a separate testing team test a complete version of the system before it is released to users.
- ✧ **User testing**, where users or potential users of a system test the system in their own environment.

5.3.1 Development Testing

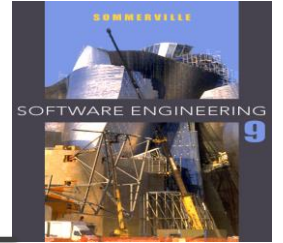


- ✧ Development testing includes all **testing activities** that are carried out by the team developing the system. 
- **Unit testing**, where individual program units or object classes are tested. Unit testing should **focus on testing the functionality of objects or methods**.
- **Component testing**, where several individual units are integrated to create composite components. Component testing should **focus on testing component interfaces**.
- **System testing**, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should **focus on testing component interactions**.
- ✧ One of the popular development testing techniques is White Box Testing (WBT) that can be applied at the unit, integration and system levels of the testing process. (WBT is discussed separately)

5.3.2 Release Testing

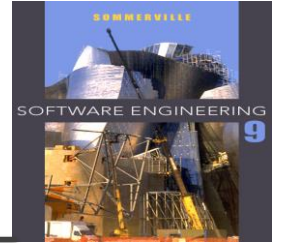
- ✧ Release testing is the process of testing a particular release of a system that is intended for use outside of the development team.
- ✧ The primary goal of the release testing process is to convince the supplier of the system that it is good enough for use.
- ✧ Release testing shows that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use.
- ✧ Release testing is usually a Black Box Testing (BBT) process where tests are only derived from the system specification. (BBT is discussed separately)

5.3.3 User Testing



- ✧ User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.
- ✧ User testing is essential, even when comprehensive system and release testing have been carried out.
 - The reason for this is that influences from the user's working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

Types of User Testing



✧ Alpha testing

- Users of the software work with the development team to test the software at the developer's site.

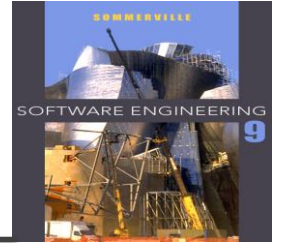
✧ Beta testing

- A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.

✧ Acceptance testing

- Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom (be-spoke) systems.

5.4 Black Box Testing (BBT)



- ✧ Black-box testing, also called behavioral testing, focuses on the functional requirements of the software.
- ✧ BBT enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.
- ✧ As shown in Figure 7.2 (next slide) BBT is concerned only with the possible inputs and their desired output regardless of the developmental details.
- ✧ BBT is performed by a separate testing team not the developer their self hence it is one type of release testing.

Black Box Testing (BBT)

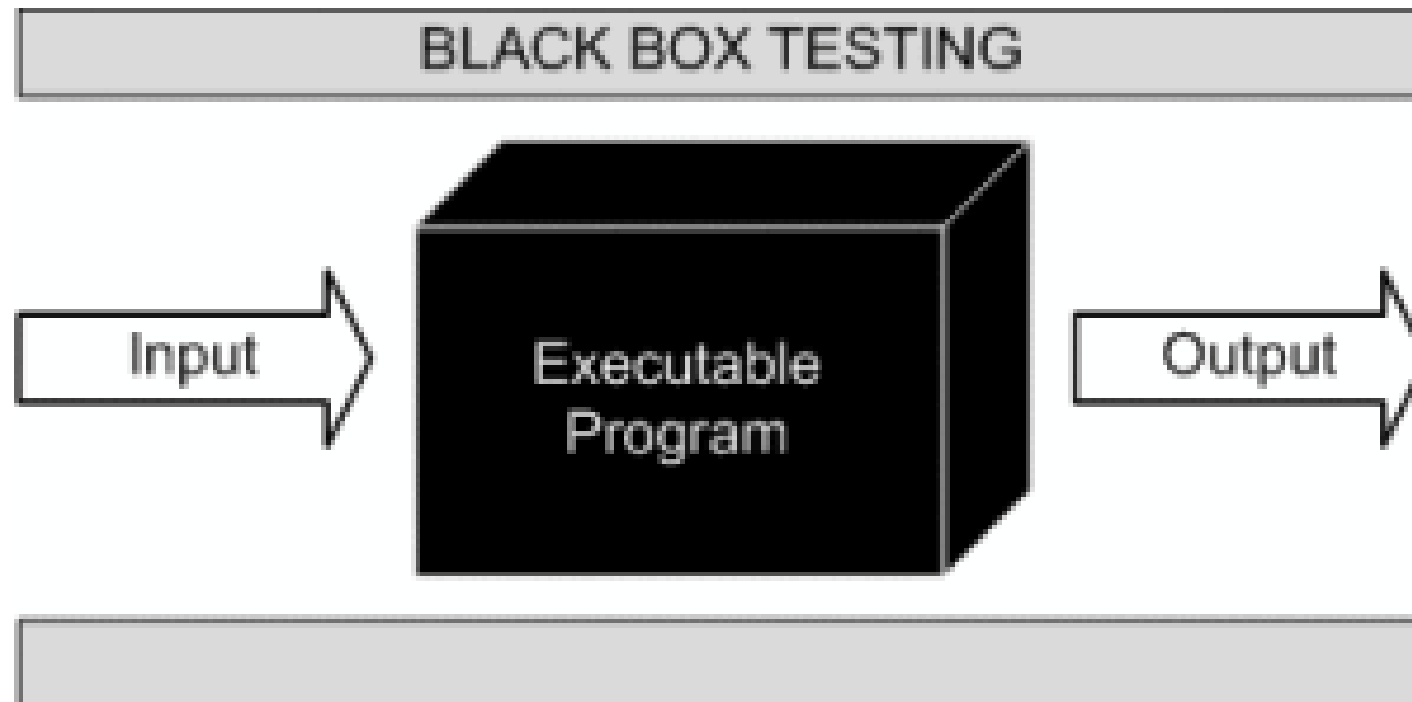
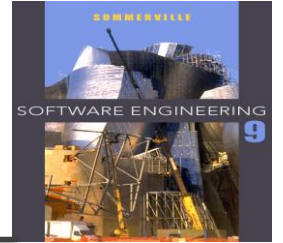
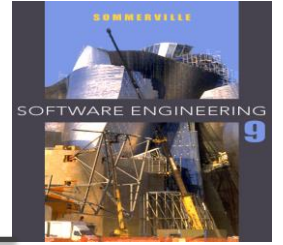


Figure 7.2: Black Box Testing

Black Box Testing (BBT)



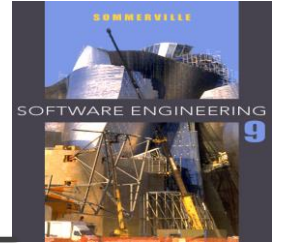
- ✧ There are further many **types of BBT** but two of the most commonly used types are as follows:
 - **Equivalence Partitioning**
 - **Boundary Value Analysis**

Equivalence Partitioning

- ✧ Equivalence partitioning is a black-box testing method that divides the input domain of a program into classes of data from which test cases can be derived.
- ✧ **Examples:** If a code of calculator has to test using BBT then possible partitioning of input test data are:

Test Cases	Possible Valid Results	Possible Invalid Results
Test case 1: $2 + 2 = ?$	$2 + 2 = 4$	$2 + 2 = 0$
Test case 2: $2.52 + 4.36 = ?$	$2.52 + 4.36 = 6.88$	$2.52 + 4.36 = 6$
Test case 3: $'a' + 345 = ?$	$'a' + 345 = \text{error}$	$'a' + 345 = 345$
Test case 4: $23 - 45 = ?$	$23 - 45 = -22$	$23 - 45 = 22$
Test case 5: $43 \times 2.4 = ?$	$43 \times 2.4 = 103.2$	$43 \times 2.4 = 86$
Test case 6: $10 / 0 = ?$	$10 / 0 = \text{error}$	$10 / 0 = 0$
Test case 7: $45 \bmod 4 = ?$	$45 \bmod 4 = 1$	$45 \bmod 4 = 11$

Boundary Value Analysis



- ✧ Boundary value analysis enhances the performance of equivalence partitioning because it leads to the selection of test cases at the "edges" of the class.
- ✧ **Examples:** In boundary value analysis testing test cases are generated as shown in Figure 7.3

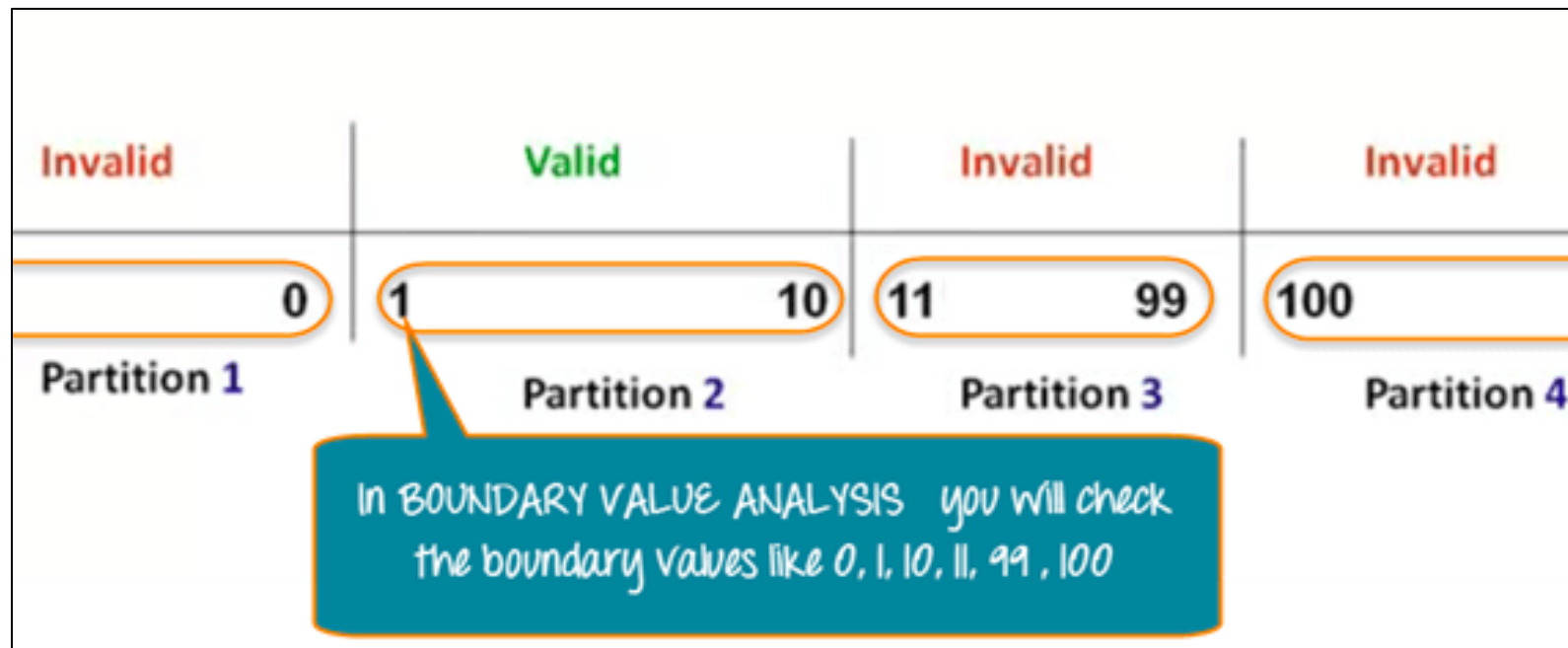
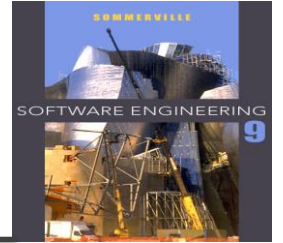


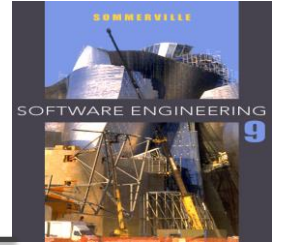
Figure 7.3: Boundary Value Analysis ranges

5.5 White Box Testing (WBT)



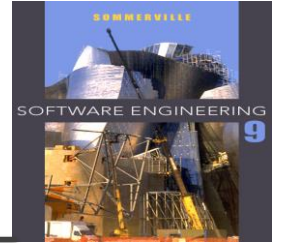
- ✧ White-box testing also called glass-box testing. In WBT test cases are designed using the control structure of the procedural design. Using white-box testing, software engineers can derive test cases that:
 - Guarantee that all independent paths within a module have been exercised at least once.
 - Exercise all logical decisions on their true and false sides
 - Execute all loops at their boundaries and within their operational bounds
 - Exercise internal data structures to ensure their validity.

White Box Testing (WBT)



- ✧ WBT can be applied at any level of development testing.
- ✧ In WBT at first code is converted into descriptive code by assigning the numbers against each significant step in the code.
- ✧ Then using the Cyclomatic complexity correctness of the program can be assured.
- ✧ **Cyclomatic complexity**: is a useful metric for predicting those modules that are likely to be error prone. It can be used for test planning as well as test case design.

Methods used in WBT



1. **Cyclomatic complexity**: corresponds to number of regions of the flow graph or possible independent paths.
2. **Cyclomatic complexity**: $V(G)$, for a flow graph, is defined as $V(G) = E - N + 2$
3. **Cyclomatic complexity**, $V(G)$, for a flow graph, is also defined as $V(G) = P + 1$

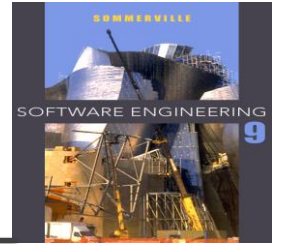
Independent path introduces at least one new set of processing statements or condition.

E is the number of flow graph edges

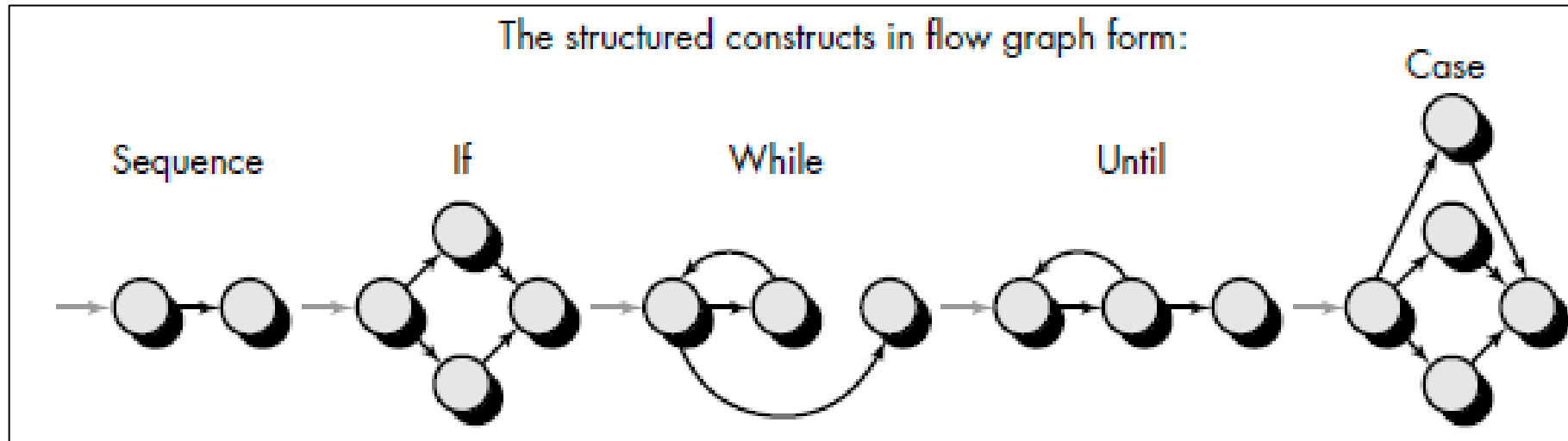
N is the number of flow graph nodes.

P is the number of predicate nodes in the flow graph.

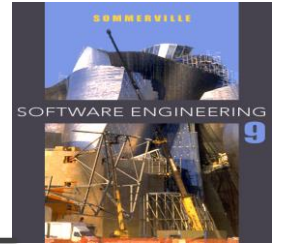
Methods used in WBT



Flow Graph construction:



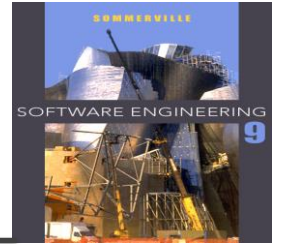
Methods used in WBT



Example-1:

Descriptive Code	FlowGraph	
<pre>y = 5 Input (value of z) } ① for(int x = 0 ; x <= z ; x = x+y) ② if(x%10==0) ③ Print("value is Even") Print("value is Multiple of Ten") } ④ else Print("value is Multiple of Five") ⑤ endif ⑥ endfor ⑦</pre>		
Cyclometric Complexity		
Possible Paths	1-2-3-4-6-2-7 1-2-3-5-6-2-7 1-2-7 Total = 3	Regions = 3
E-V+2	8 - 7 + 2 = 3	
No of Predicate Nodes + 1	2 + 1 = 3	

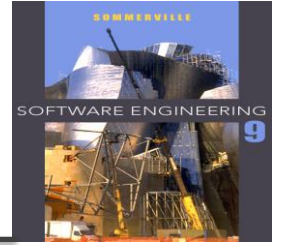
Methods used in WBT



Example-2

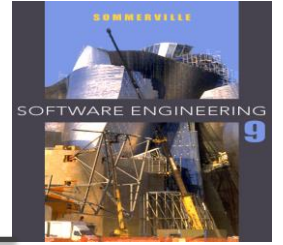
Descriptive Code	FlowGraph	
<div><pre>A[3] = {1, 3, 5} B[3] = {2, 4, 6} C[3] for(i = 0, i < 3, i ++) C[i] = A[i] + B[i] end-for for(i = 0, i < 3, i ++) Print(C[i]) end-for</pre></div>	<pre>graph TD 1((1)) --> 2((2)) 2 -- R1 --> 3((3)) 3 --> 4((4)) 4 -- R3 --> 5((5)) 5 -- R2 --> 6((6)) 6 --> 7((7))</pre>	
Cyclometric Complexity		
Possible Paths	<div>1-2-3-2-4-5-6-5-7</div> <div>1-2-4-5-6-5-7</div> <div>1-2-4-5-7</div> <div>Total = 3</div>	Regions = 3
E-V+2	8 - 7 + 2 = 3	
No of Predicate Nodes + 1	2 + 1 = 3	

5.6 Software Maintenance



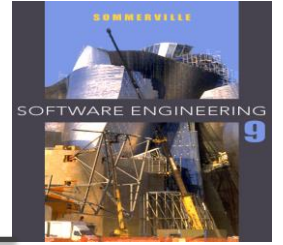
- ✧ Modifying a program after it has been put into use.
- ✧ The term is mostly used for changing custom software. Generic software products are said to evolve to create new versions.
- ✧ Maintenance does not normally involve major changes to the system's architecture.
- ✧ Changes are implemented by modifying existing components and adding new components to the system.

5.7 Problems During Maintenance



1. Often the program is written by another person or group of persons.
2. Often the program is changed by person who did not understand it clearly.
3. Program listings are not structured.
4. High staff turnover.
5. Information gap.
6. Systems are not designed for change

5.8 Types of Maintenance



- ✧ Maintenance to repair software faults
 - Changing a system to correct deficiencies in the way meets its requirements.
- ✧ Maintenance to adapt software to a different operating environment
 - Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation.
- ✧ Maintenance to add to or modify the system's functionality
 - Modifying the system to satisfy new requirements.

5.9 Software Maintenance Process

- ✧ Once particular maintenance objective is established, the maintenance personnel must first understand what they are to modify.
- ✧ They must then modify the program to satisfy the maintenance objectives.
- ✧ After modification, they must ensure that the modification does not affect other portions of the program.
- ✧ Finally, they must test the program. These activities can be accomplished in the four phases as shown in Figure 8.2.

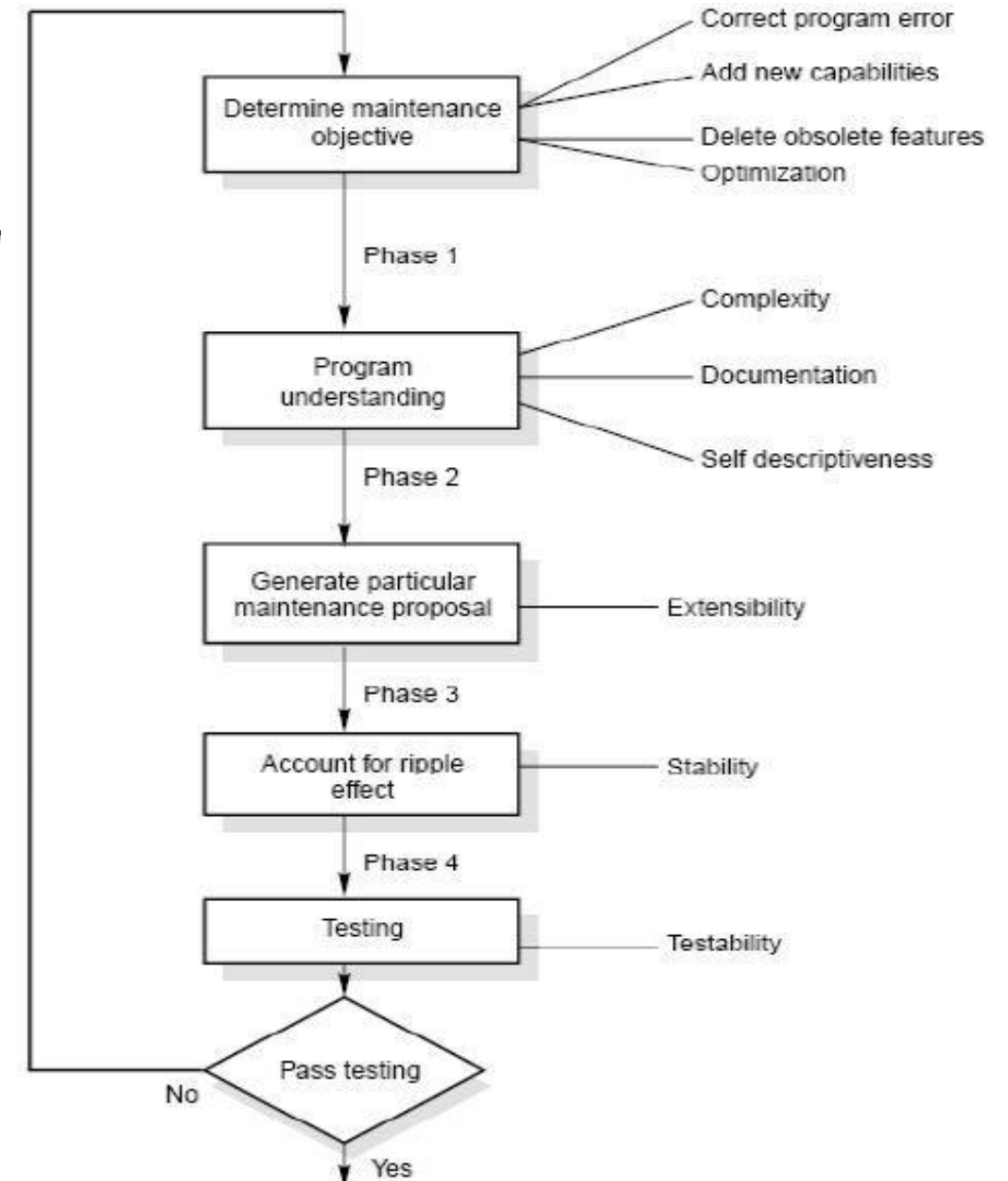


Figure 8.2: Software Maintenance Process

5.10 Maintenance Costs

- ✧ Usually greater than development costs (2 to 100 times 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 and Maintenance Costs

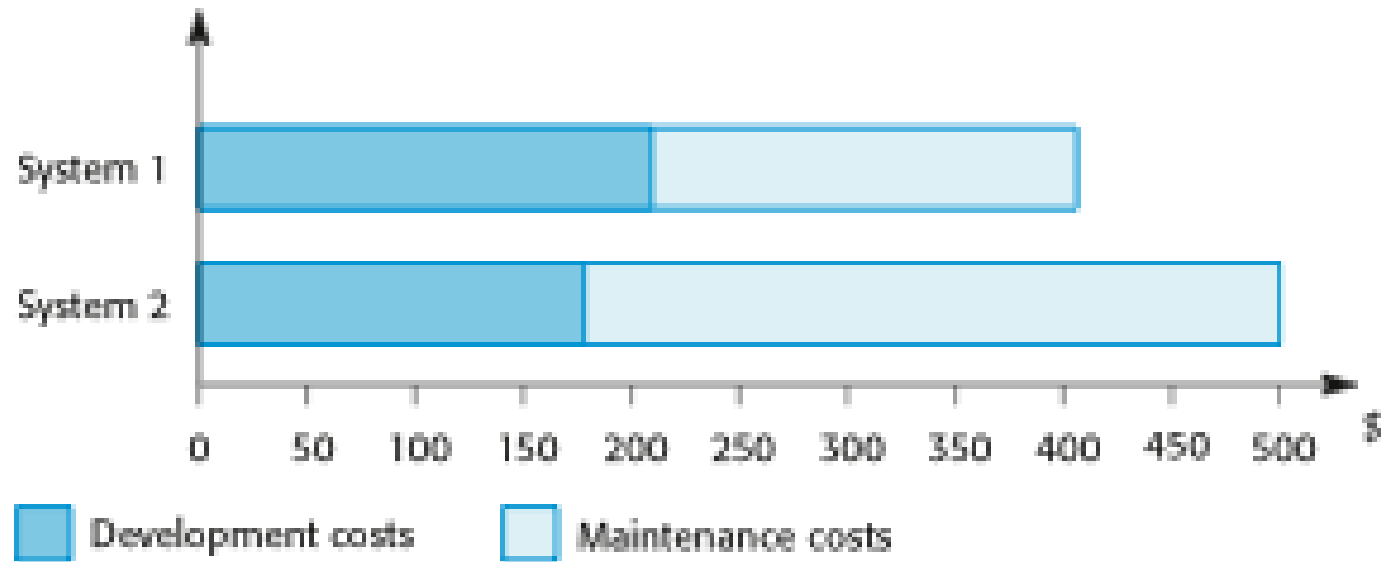
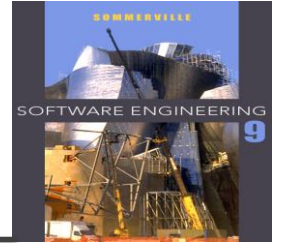


Figure 8.3: Development and Maintenance Costs

5.10.1 Maintenance Cost Factors

✧ Team stability

- Maintenance costs are reduced if the same staff are involved with the organization.

✧ Contractual responsibility

- The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change.

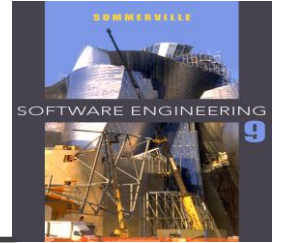
✧ Staff skills

- 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.

5.11 Maintenance Prediction



- ✧ Maintenance prediction is concerned by identifying the parts of the system that may cause problems.
- ✧ More and more changes degrades the system and reduces its maintainability.

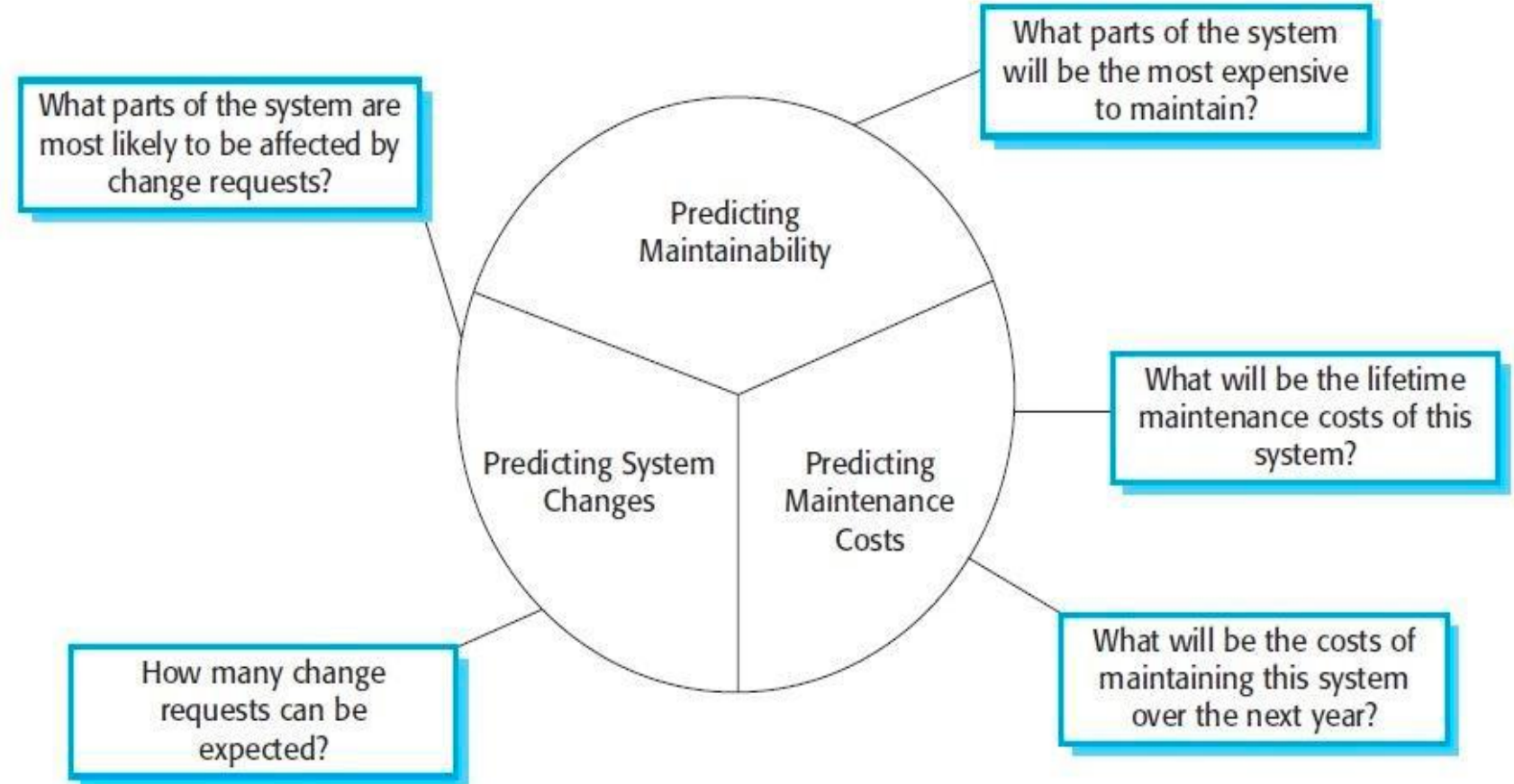


Figure 8.4: Maintenance Prediction

5.12 Software Re-Engineering

Self
req

- ✧ Software re-engineering is concerned with taking existing legacy systems and re-implementing them to make them more maintainable without changing its functionality.
- ✧ The critical distinction between re-engineering and new software development is the starting point for the development as shown in Figure 8.5.

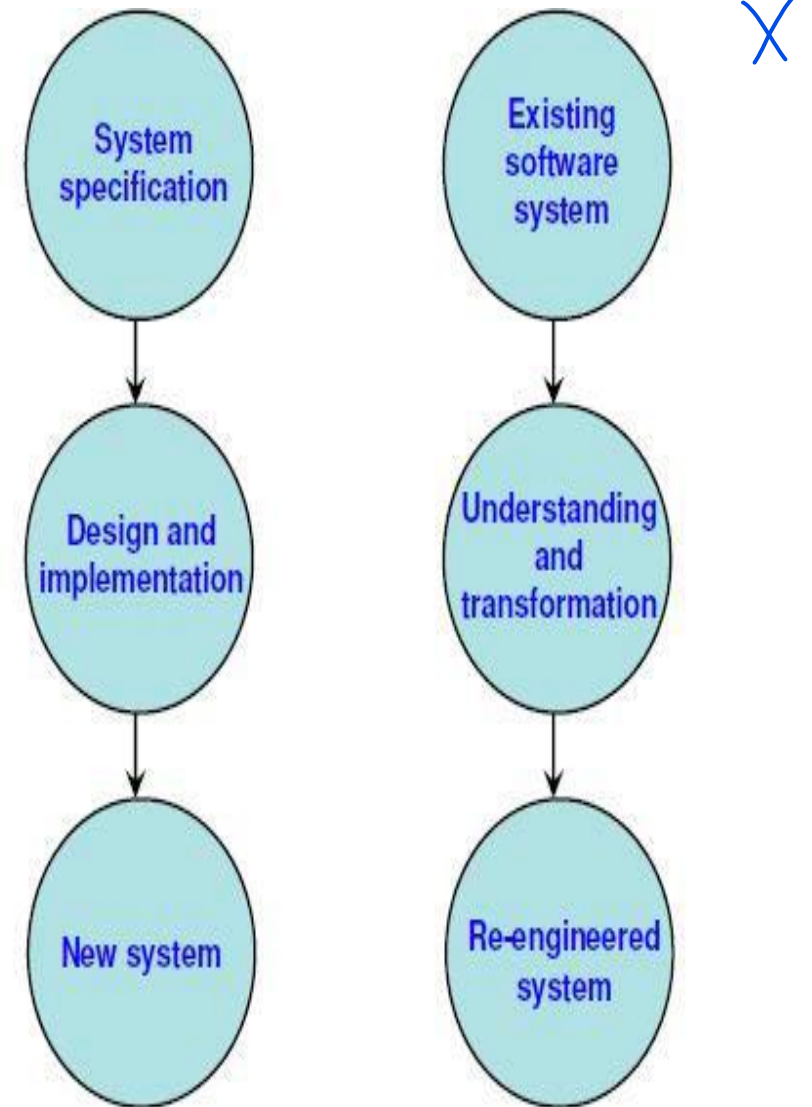
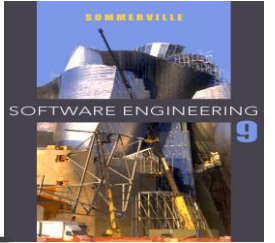


Figure 8.5: Comparison of new software development with re-engineering

Self
(m29)

X

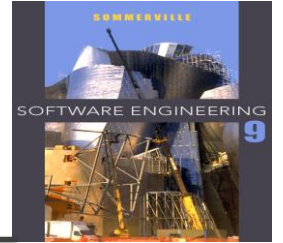
Software Re-Engineering



- ✧ As a part of this re-engineering process, the system may be re-documented or restructured.
- ✧ It may be translated to a more modern programming language, implemented on existing hardware technology. Thus software re-engineering allows translate source code to new language, restructured old code, migrate to a new platform (such as client-server), capture and then graphically display design information, and re-document poorly documented systems.
- ✧ The cost of re-engineering depend on the extent of the work that is carried out. Other factors affecting costs are: the quality of the software, tool support available, extent of data conversion, availability of expert staff.
- ✧ The alternative to re-engineering a software system is to develop that system using modern software engineering techniques. Where systems are very badly structured this may be the only viable option as the re-engineering costs for these systems are likely to be high.

5.12.1 Advantages of RE-Engineering

Self
(mcq)



✧ Reduced risk

- There is a high risk in new software development. There may be development problems, staffing problems and specification problems.

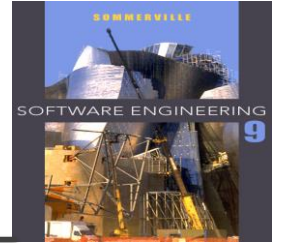
✧ Reduced cost

- The cost of re-engineering is often significantly less than the costs of developing new software.

5.12.2 Reengineering process activities

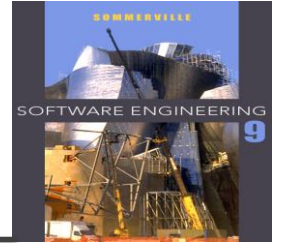
- ✧ Source code translation
 - Convert code to a new language.
- ✧ Reverse engineering
 - Analyse the program to understand it;
- ✧ Program structure improvement
 - Restructure automatically for understandability;
- ✧ Program modularisation
 - Reorganise the program structure;
- ✧ Data reengineering
 - Clean-up and restructure system data.

5.12.3 Reengineering cost factors



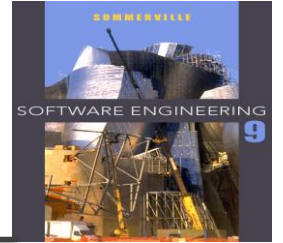
- ✧ The quality of the software to be reengineered.
- ✧ The tool support available for reengineering.
- ✧ The extent of the data conversion which is required.
- ✧ The availability of expert staff for reengineering.
 - This can be a problem with old systems based on technology that is no longer widely used.

Key Points



- ✧ software application needs to be verified as well as validated for a successful deployment.
- ✧ Testing is the process of executing a program with the intent of finding errors.
- ✧ **Stages of Testing**-Development testing, Release testing and User testing
- ✧ Black-box testing, also called behavioral testing, focuses on the functional requirements of the software.
- ✧ White-box testing also called glass-box testing. In WBT test cases are designed using the control structure of the procedural design.

Key Points



- ✧ There are 3 types of software maintenance, namely bug fixing, modifying software to work in a new environment, and implementing new or changed requirements.
- ✧ Software re-engineering is concerned with re-structuring and re-documenting software to make it easier to understand and change.
- ✧ The business value of a legacy system and the quality of the application should be assessed to help decide if a system should be replaced, transformed or maintained.