CZ3002 - Advanced Software Engineering

Software Maintenance

Faculty	:	Dr Althea Liang
School	:	School of Computer Science and Engineering
Email	:	qhliang@ntu.edu.sg
Office	:	N4-02c-107

Quick Review Sheet of Previous Lesson

Knowledge blocks and skills obtained in the last lesson:

- Outline the **design ideas for change (maintainability)**
- Understand design ideas for change in form of **design patterns**
  - Definition and purposes of **design patterns**
  - Architectural patterns and mid-level design patterns
  - **Model-View-Controller (MVC)** Application Framework and Quick discussion on **service-oriented design** and **benefits**.

Lesson Objectives

At the end of the lesson, you should be able to:

- ▶ Describe different types of maintenance and understand cost of maintenance, with the help of examples
- ▶ Explain evolution of software systems
- ▶ List the key issues in software maintenance
- ▶ Explain Commercial-Off-The-Shelf (COTS) Components
- ▶ Apply techniques of re-engineering and reverse engineering

Preview: Example of Organisation in this Lesson

Examples	Slides
System 1 and System 2	Development vs. Maintenance Costs
Example E	Nature of Maintenance
Example SQ	System Quality and Business Value

Preview: Best Practices of an Organisation in this Lesson

Best practices/ Tools	Slides
Laws	Lehman’s Laws

SWEBOK Software Maintenance Knowledge Area

What is software maintenance about?

Fundamentals

- Nature of Maintenance
- Software Evolution

Kinds of Maintenance Processes

Key Issues

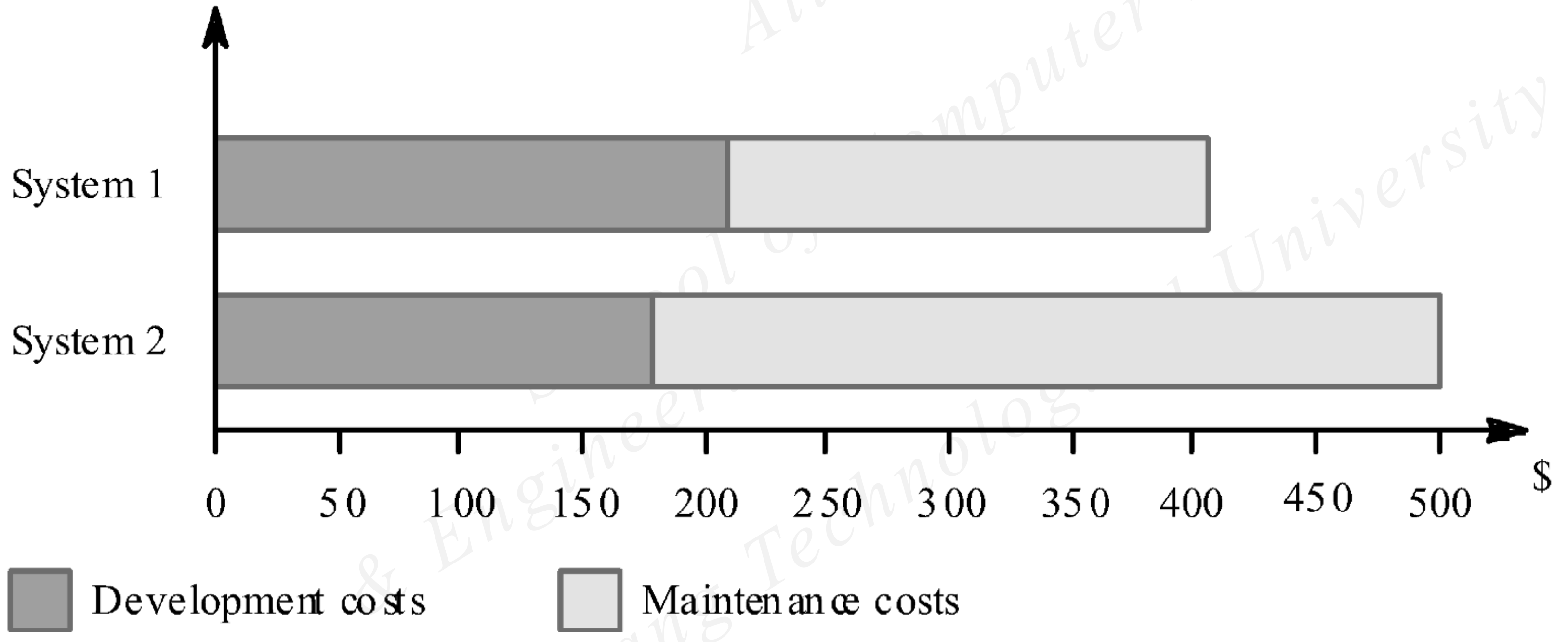
- Cost
- Designing for Maintainability

Techniques

- Reverse Engineering
- Re-Engineering

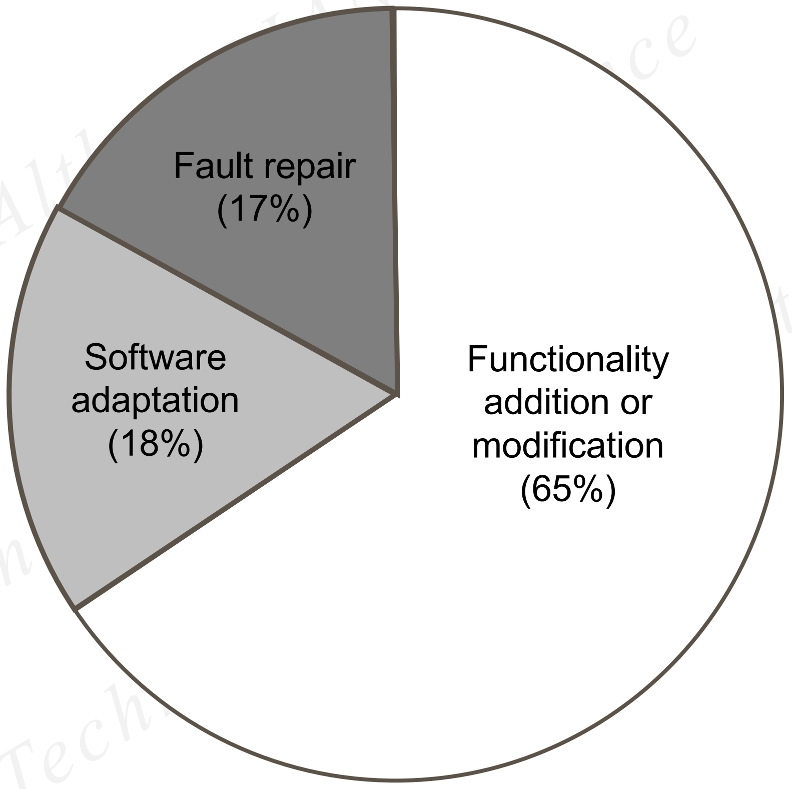
Development vs. Maintenance Costs (System1, System2)

- Any work done to the system after it is in operation is considered maintenance.
- On the average, 20% of effort goes into development and 80% into maintenance.



Nature of Maintenance (Example E)

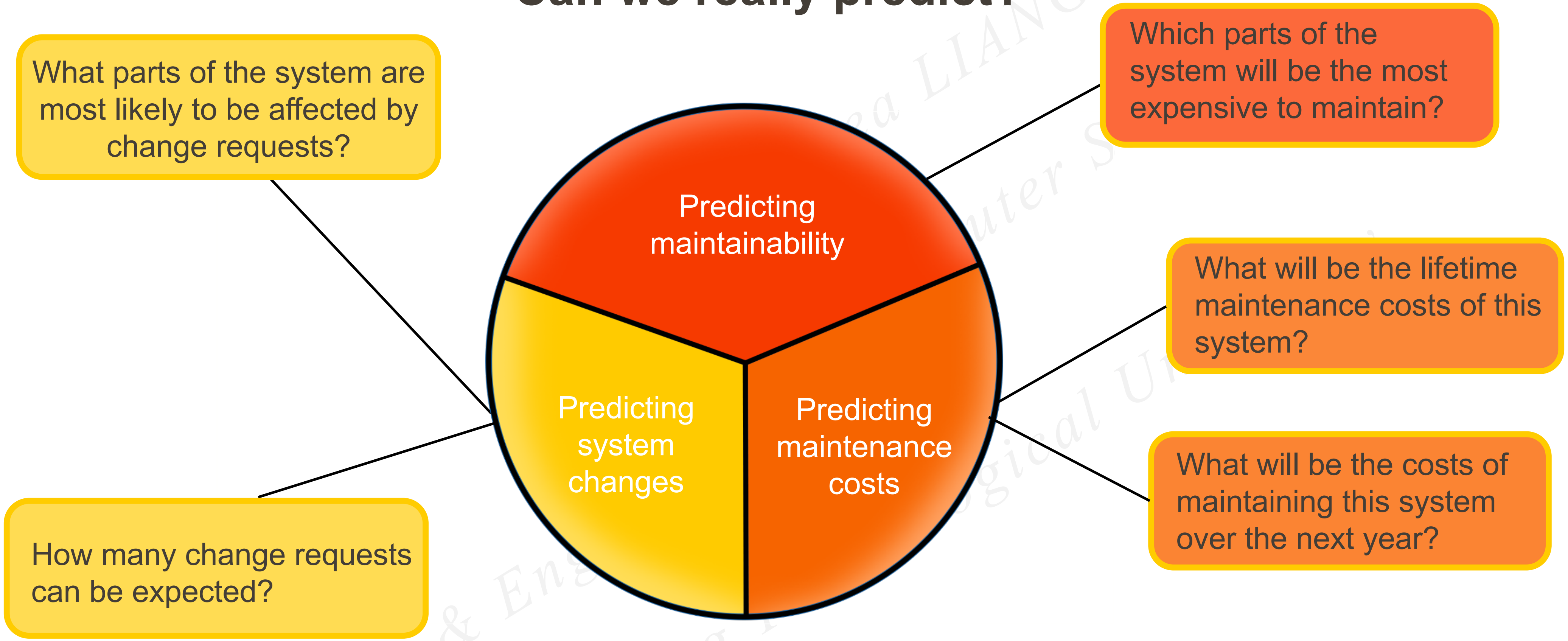
- Correction
  - Corrective
  - Preventive
- Enhancement
  - Adaptive
  - Perfective
- Metrics
  - Defect Rate
  - Mean Time to Change



Maintenance Effort Distribution

Maintenance Prediction and Prediction Model

Can we really predict?


What parts of the system are most likely to be affected by change requests?
How many change requests can be expected?
Predicting maintainability
Predicting system changes
Predicting maintenance costs
Which parts of the system will be the most expensive to maintain?
What will be the lifetime maintenance costs of this system?
What will be the costs of maintaining this system over the next year?

NANYANG TECHNOLOGICAL UNIVERSITY

Lehman’s Laws

Law	Description
Continuing change	A program that is used in a real-world environment must change or become progressively less useful in that environment.
Increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
Declining quality	The quality of systems will appear to be declining unless they adapt to changes in their operational environment.
Organisational stability	Over a program’s lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.

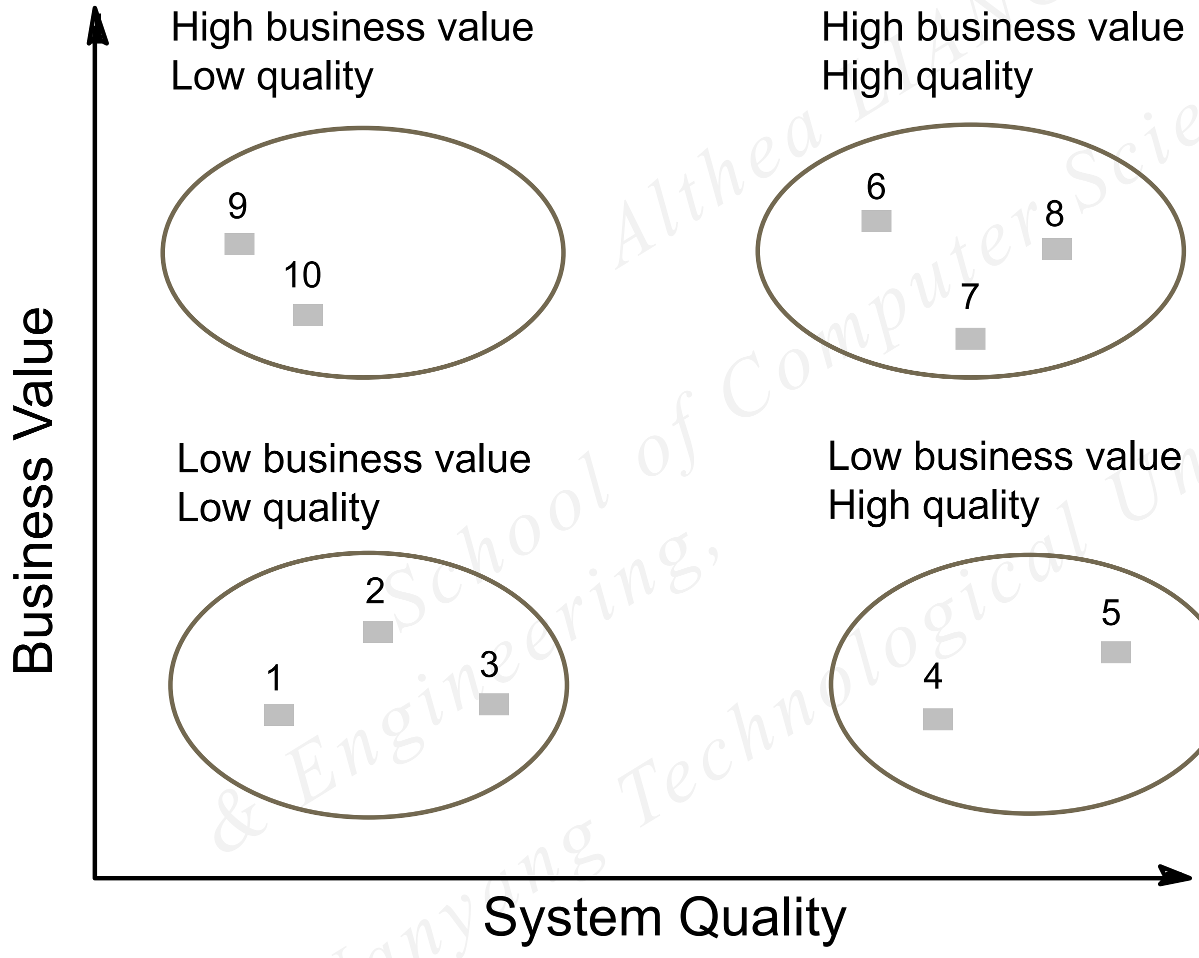
Importance of Evolution

- Organisations have huge investments in their software systems - they are critical business assets.
- To maintain the value of these assets to the business, they must be changed and updated.
- The majority of the software budget in large companies is devoted to evolving existing software rather than developing new software.

Legacy System Evolution

- Organisations that rely on legacy systems must choose a strategy for evolving these systems.
  - Scrap the system completely and modify business processes so that it is no longer required.
  - Continue maintaining the system.
  - Transform the system with re-engineering to improve its maintainability.
  - Replace the system with a newer system.
- The strategy chosen should depend on the system quality and its business value.

System Quality and Business Value (Example SQ)



Key Issues in Maintenance

- Technical
  - Limited understanding
  - Test coverage
  - Poor maintainability to begin with
- Management
  - Staffing – morale, skillset
  - Contractual responsibility – service level agreement
- Cost
- Measurement

Commercial-Off-The-Shelf (COTS) Components

- Building large systems by integrating COTS systems is now a viable development strategy for some types of system such as E-commerce systems.

What are COTS Components?

- Usually complete application systems that offer an API (Application Programming Interface).

Key Benefits

- Faster application development
- Usually lower development costs

Design Decisions with COTS

- Which COTS products offer the most appropriate functionality?
  - There may be several similar products that may be used.
- How will data be exchanged?
  - Individual products use their own data structures and formats.
- What features of the product will actually be used?
  - Most products have more functionality than is needed. You should try to deny access to unused functionality.

Maintenance Considerations with COTS

- Lack of control over functionality and performance
  - COTS systems may be less effective than they appear.
- Problems with COTS system inter-operability
  - Different COTS systems may make different assumptions that means integration is difficult.
- No control over system evolution
  - COTS vendors, not system users, control evolution.
- Support from COTS vendors
  - COTS vendors may not offer support over the lifetime of the product.

System Re-Engineering

What is System Re-Engineering?

- Re-structuring or re-writing part or all of a legacy system without changing its functionality.
- Involves adding effort to make the sub-systems easier to maintain.
- The system may be re-structured and re-documented.

When is it applicable?

- Where some (but not all) sub-systems of a larger system require frequent maintenance.

Re-Engineering Process

- Reverse Engineering
  - Analyse software to identify components and their relationships
  - Create representation at higher levels of abstraction e.g. documentation, UML diagrams, ER diagrams
- Forward Engineering
  - Programme structure improvement
  - Programme modularisation
  - Data re-engineering

Re-Engineering Cost Factors

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

Summary

Now you should be able to:

- Describe different types of maintenance and understand cost of maintenance, with the help of examples
- Explain evolution of software systems
- List the key issues in software maintenance
- Explain Commercial-Off-The-Shelf (COTS) Components
- Apply techniques of re-engineering and reverse engineering

NANYANG TECHNOLOGICAL UNIVERSITY

Special Thanks to Kydon during the TEL Efforts of the Lecture

End of Software Maintenance

Faculty	:	Dr Althea Liang
School	:	School of Computer Science and Engineering
Email	:	qhliang@ntu.edu.sg
Office	:	N4-02c-107