Software Evolution

Chapter 11

Table of Contents

a) Introduction - Software Evolution Process
b) Software Maintenance
c) Software Reengineering & Reverse Engineering

a) Introduction - Software Evolution Process

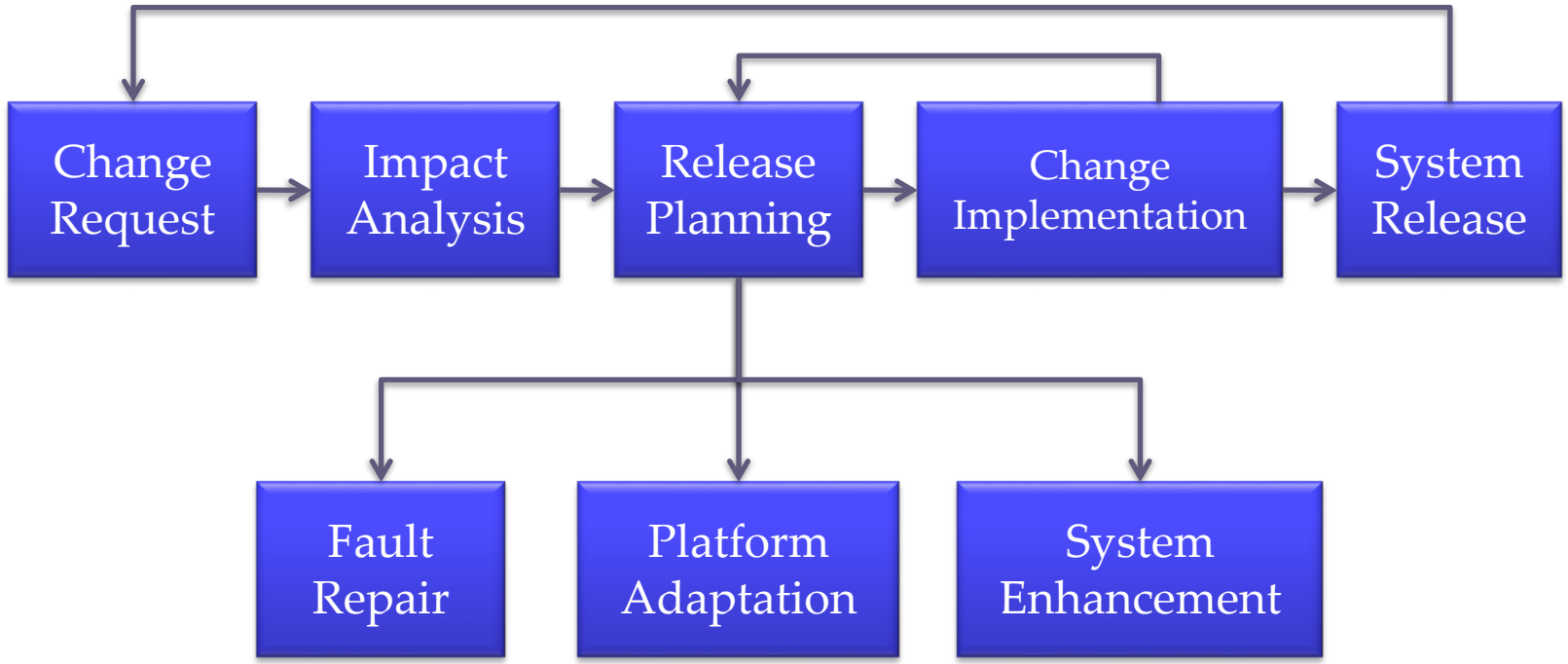
a) Introduction – Software Evolution Process

- Vary considerably depending on the type of software being maintained, the development processes used and the people involved
- Can be formal or informal process
- System change proposals are the driver for system evolution

a) Introduction – Software Evolution Process

- Fundamental activities:
  - Change analysis
  - Release planning
  - System implementation
  - System release

a) Introduction – Software Evolution Process



b) Software Maintenance

Lesson Objectives

- Explain various types of software maintenance
- Factors affect maintenance costs

Software Maintenance

- The process of changing a system after it has been delivered and is in use is called software maintenance.

- Some reasons for changes:
  - Software errors
  - Installation of new hardware
  - Customer needs

Types of Software Maintenance

i. ***Corrective maintenance*** that concerns with fixing reported errors in the software.

ii. ***Adaptive maintenance*** that requires changing the software to some new environment such as a different hardware platform or for use with a different operating system

iii. ***Perfective maintenance*** involves implementing new functional or non-functional system requirements. These are generated by software customers as their organization or business changes

Factors that Affecting Maintenance Cost

Module independence

- ✓ High independency, low cost

Factors that Affecting Maintenance Cost

Programming Language

- ✓ High level programming language

Factors that Affecting Maintenance Cost

Programming Style

- ✓ Good structure → low cost

Factors that Affecting Maintenance Cost

Program Validation & Testing

- ✓ More time spent in testing, ↓error, ↓cost

Factors that Affecting Maintenance Cost

Documentation Quality

- ✓ Good quality, good understanding, ↓cost

Factors that Affecting Maintenance Cost

Configuration Management Techniques

- ✓ Effect technique → easy to keep track all versions, ↓cost

Factors that Affecting Maintenance Cost

Application Domain

- ✓ New application domain, less understanding, ↑cost

Factors that Affecting Maintenance Cost

Staff Stability

- ✓ New project reassignment, ↑cost
- ✓ Same staff maintain, ↓cost

Factors that Affecting Maintenance Cost

Age of the program

- ✓ > older, > maintenance it receives, > cost

Factors that Affecting Maintenance Cost

The dependence of the program on its external environment

- ✓ > depend on environment changes, > cost

Factors that Affecting Maintenance Cost

Hardware stability

- ✓ If hardware no need to change, < cost (rarely happen)

Maintenance Cost

The costs of adding functionality to a system after it has been put into operation are usually much greater than providing similar functionality when software is originally developed.

WHY?

Maintenance Cost

- Maintenance staff are often relatively inexperienced and unfamiliar with the application domain
- The programs being maintained may have been developed many years ago without modern software engineering techniques. Therefore they may be unstructured and difficult to understand.
- Changes made to a program may introduce new faults, which trigger further change requests.

Maintenance Cost

- As a system is changed, its structure tends to degrade. This makes the system harder to understand & the program becomes less cohesive.
- The links between a program and its associated documentation are sometimes lost during the maintenance process. The documentation may therefore be an unreliable aid to program understanding.

c) Software Reengineering

Lesson Objectives

- What is software reengineering
- Differentiate software reengineering to reverse engineering
- The advantages and disadvantages of software reengineering
- Legacy system management

Introduction

- In some businesses, it has been estimated that 80% of all software expenditure is consumed by system maintenance and evolution (Yourdon, 1989)
- This is particularly true for legacy system
- In 1990, it was estimated (Ulrich, 1990) that there were 120 billion lines of source code of legacy system!

Introduction

- Legacy systems are system, which have been in existence for some time and which are essential for the successful functioning of an organization.
- Why do they still exist?
  - Business procedures & knowledge of system may not documented elsewhere
  - Hence, too risky to rewrite them

Introduction

- Most legacy systems were developed before software engineering techniques were widely used.
- Therefore …
  - The system may be poorly structured
  - Documentation may be out-of-date
  - Documentation may be inconsistent
  - Etc.
- Developers can be long gone!

Introduction

- Solution to make legacy systems more maintainable?

Software Reengineering Perhaps

Introduction

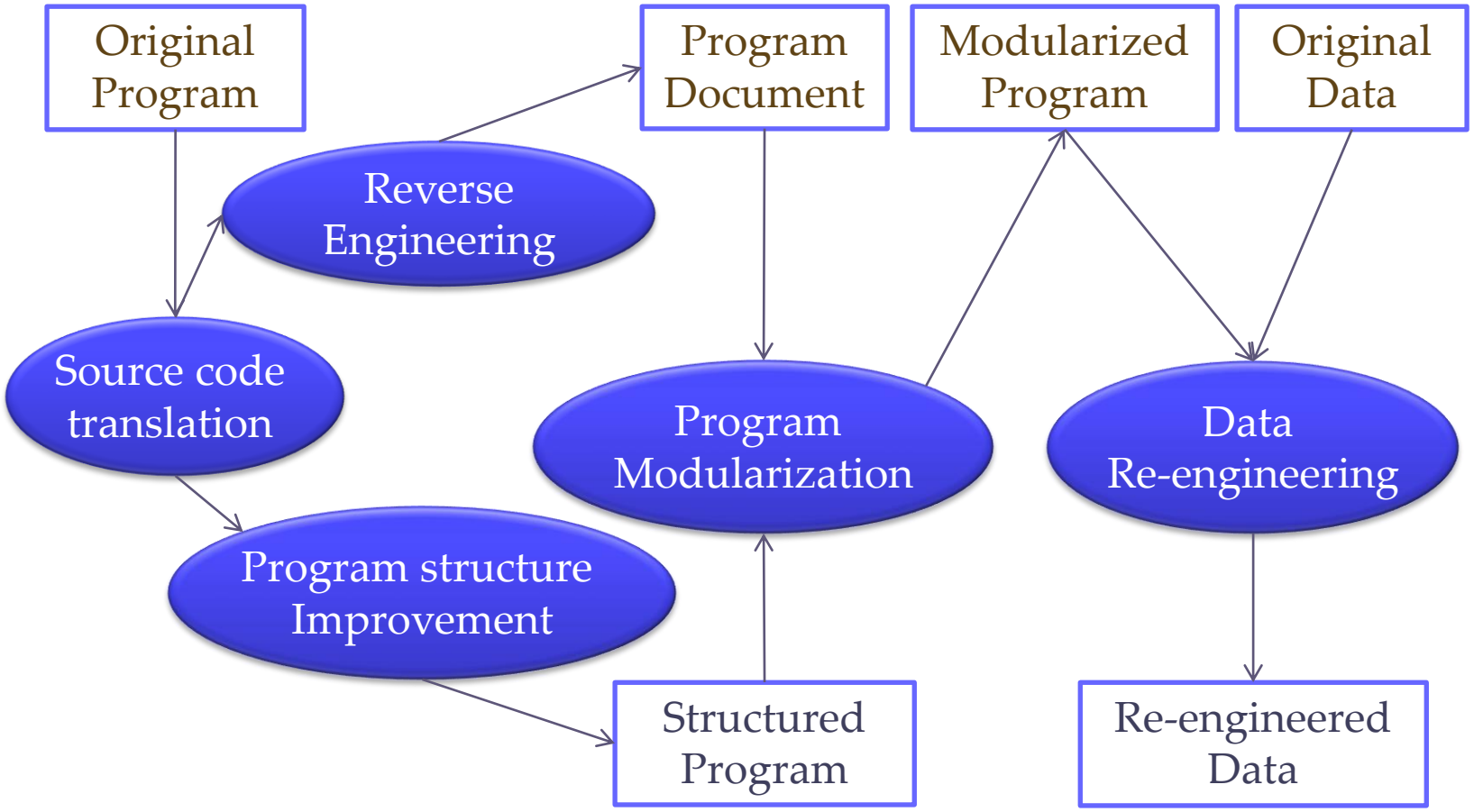
- The main objectives of system re-engineering are to:
  - improve the system structure
  - create new system documentations
  - make it easier to understand.

  So, the cost of future system maintenance should therefore be reduced.
- In other words, software re-engineering is concerned with taking existing legacy systems and **re-implementing** them to make them more maintainable.

How?

- **Source code translation**
- **Reverse Engineering**
- **Program restructuring**
- **Program modularization**
- **Data reengineering**

Software Re-engineering Process



Software Re-engineering Process
1. Source Code Translation

- The simplest form of software re-engineering is program translation where source code in one programming language is translated to source code in some other language (Fortran → C).
- Source level translation may be needed for the following reasons: -
  - **hardware platform update**: compilers for original language may not work on new hardware
  - **staff skill shortages**: maintenance staff do not know the obsolete language
  - **organizational policy changes**: standardize on a few languages to minimize support cost

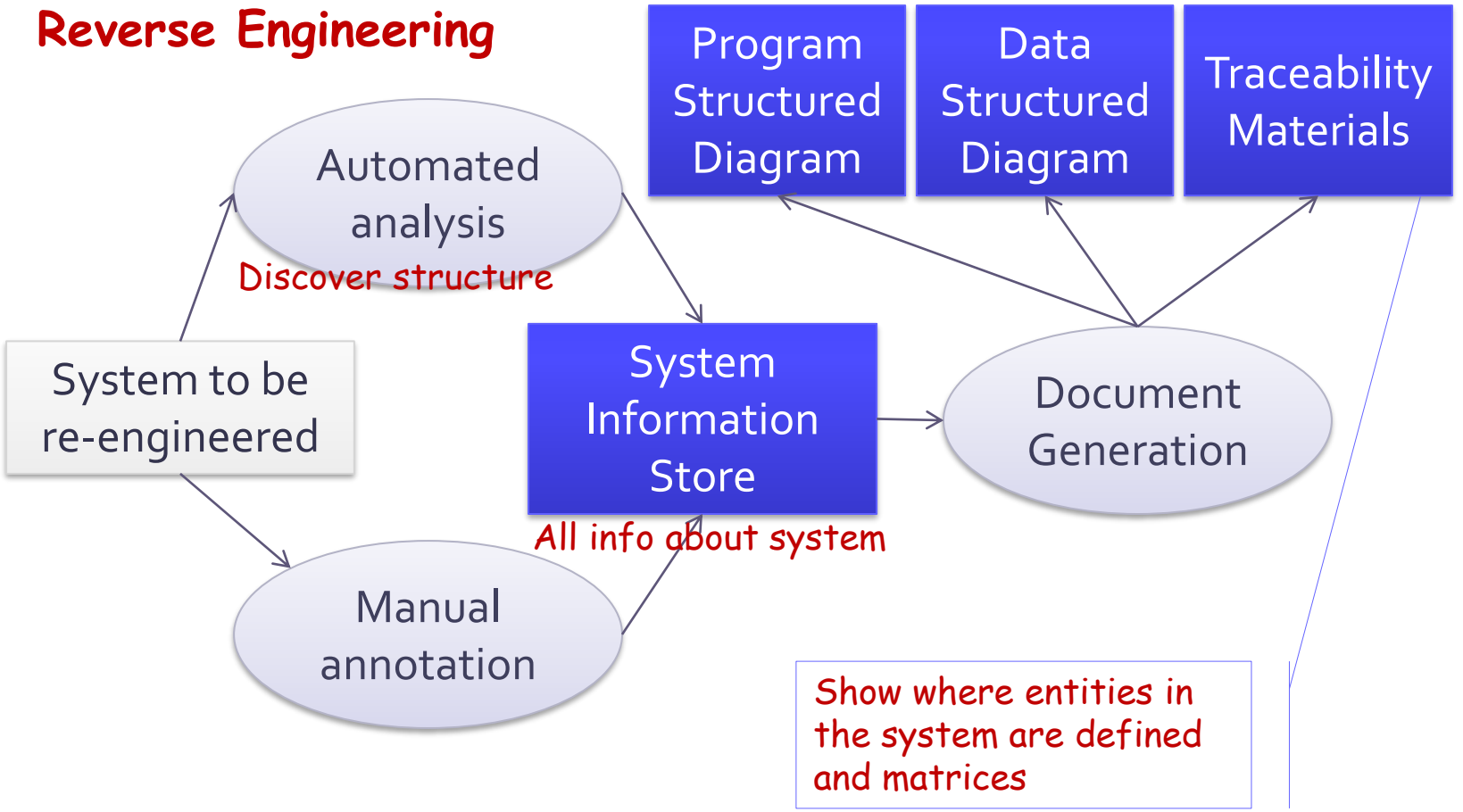
Software Re-engineering Process
1. Source Code Translation

- Economically unrealistic unless automatic translator (CASE tools) is available
- REFINE system has powerful pattern matching and program translation capabilities
- Some manual work is still needed though

Software Re-engineering Process
2. Reverse Engineering

- Reverse engineering is the process of deriving a system's design and specification from its source code.
- In other words, reverse engineering is the process of analyzing software with the objective of recovering its design and specification (documentation).
- Reverse engineering is normally part of s/w re-engineering process.
- The design & specification are recovered to help understand a program before reorganizing/ modularizing its structure.

Software Re-engineering Process



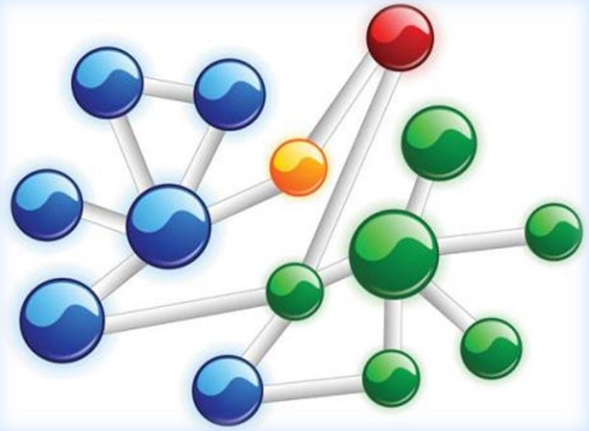
Software Re-engineering Process
2. Reverse Engineering

Further more,

- The design and specification of an existing system may be reverse engineered so that they can serve as an input to the requirements specification for that program's replacement.
- Alternatively, the design and specification may be reverse engineered so that they are available to help program maintenance. With this additional information, it may not be necessary to re-engineer the system source code.

Software Re-engineering Process
3. Program Structure Improvement

- The control structure of the program is analysed and modified
- E.g. Restructuring/replacing 'spaghetti' program structure such as ***gotos*** with ***if-then-else*** conditions and ***while*** loops
- Complex conditions can also be simplified
- This makes a program more readable and easier to understand i.e. more maintainable



```
Start:      Get (Time-on, Time-off, Setting, Temp, Switch)
            if Switch = off goto off
            if Switch = on goto on
            goto CntrID
off:        if Heating-status = on goto Sw-off
            goto loop
on:         if Heating-status = off goto Sw-on
            goto loop
CntrId:     if Time = Time-on goto on
            if Time = Time-off goto off
            if Time < Time-on goto Start
            if Time > Time-off goto Start
            if Temp > Setting then goto off
            if Temp < Setting then goto on
Sw-off:     Heating-status := off
            goto Switch
Sw-on:      Heating-status := on
Switch:     Switch-heating
loop:       goto Start
```

A control program with spaghetti logic

```
Loop
        Get (…)
        Case      Switch off
                          When    On ->
                          When    Off →
                          When    Controlled →
        End Case
End Loop
```

a structured control program

Example of Condition Simplification

Complex condition

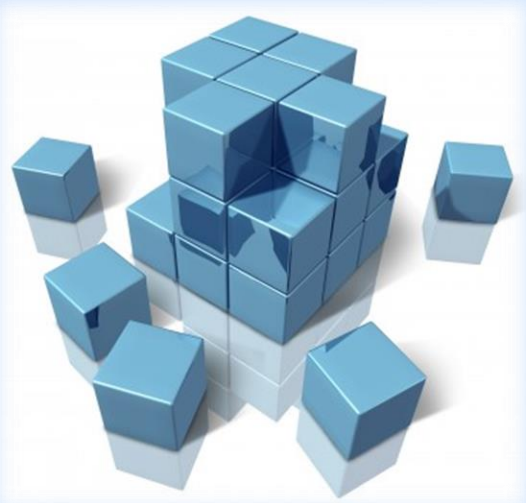
```
if not (A > B and (C < D or not (E > F))) …
```

Simplified condition

```
if A <= B and (C >= D or E > F)…
```

Software Re-engineering Process
4. Program Modularization

- Related parts of the program are grouped together and redundancy is removed.
- May involve architectural transformation where centralised system may modified to run on a distributed platform



Software Re-engineering Process
5. Data Re-engineering

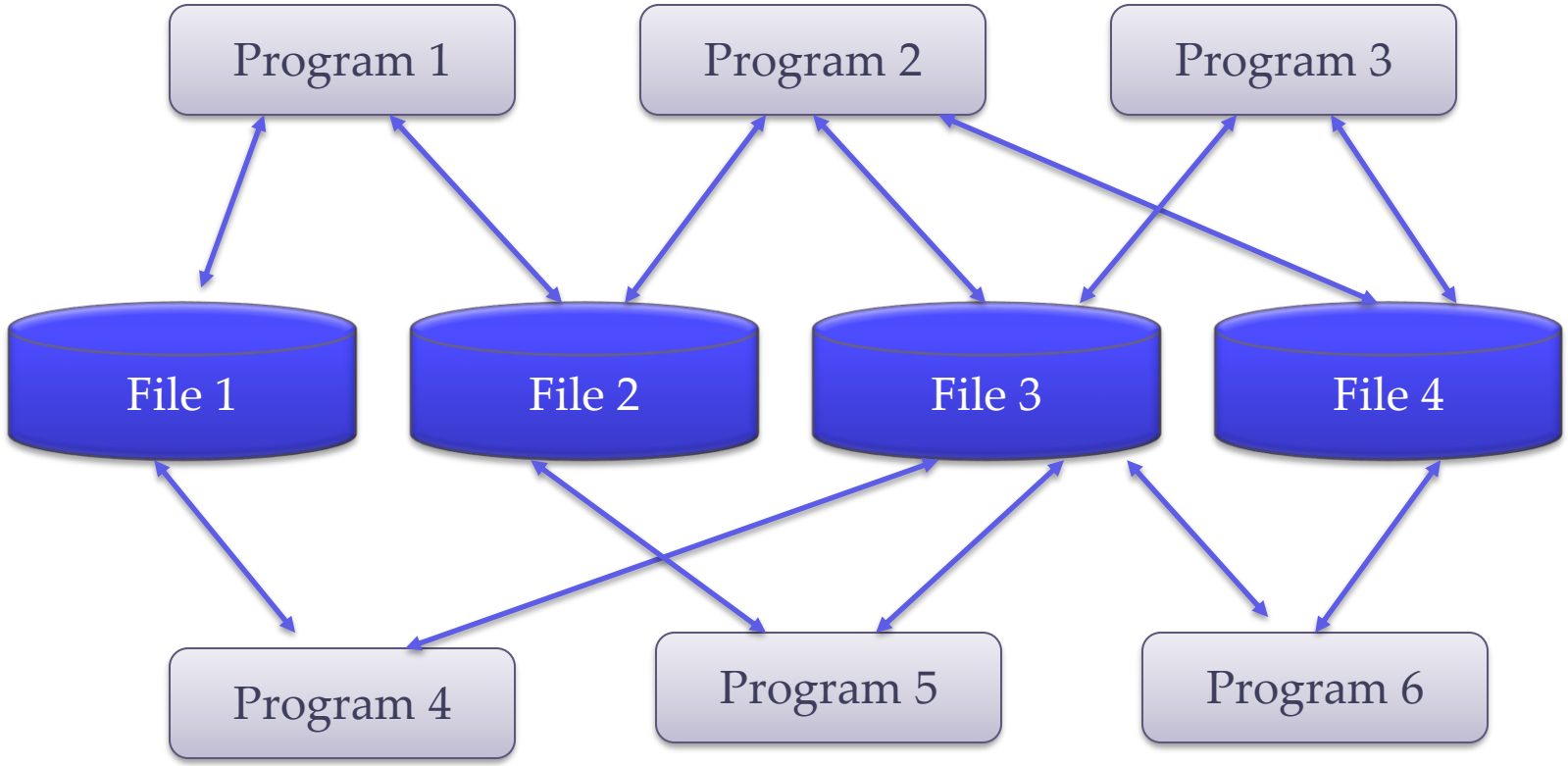
- How to re-engineer? (refer diagram)
- Some of the problems (b4 data re-engineering) with data which can arise in legacy systems made up of several cooperating programs are: -
  - data naming (for files/attributes) problems
  - field length problems: assigned diff length in diff program!
  - record organization problems : same entity may be org differently in diff programs, e.g. COBOL language.
  - hard-coded literals/values: such as tax rate
  - no data dictionary: to define their representation etc.

Software Re-engineering Process
5. Data Re-engineering

- Data values may also be stored in an inconsistent way. After the data definitions have been re-engineered, the data values must also be converted to conform to the new structure. Some of the data value inconsistencies problems (before data re-engineering) are:
  - Inconsistent default values: different date format?
  - Inconsistent validation rules
  - Inconsistent units: kg vs pound
  - Inconsistent representation semantic
  - Inconsistent handling of negative values

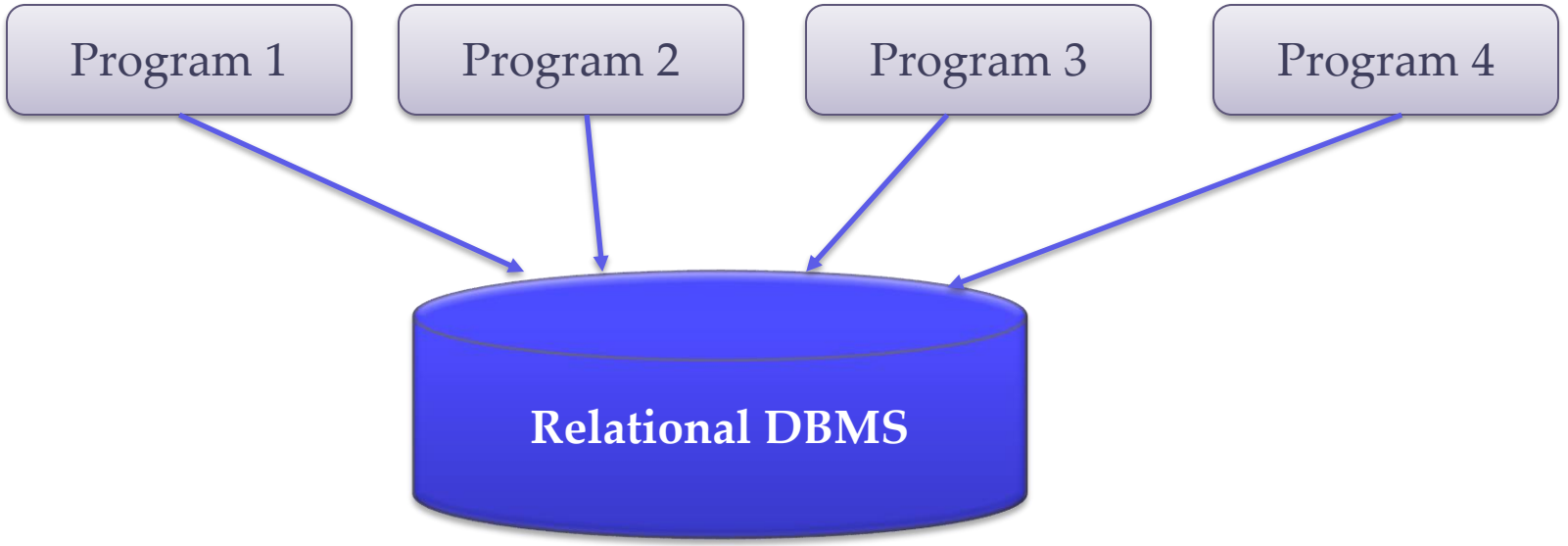
Software Re-engineering Process

Data Re-engineering (Before)



Software Re-engineering Process

Data Re-engineering (After)



Software Re-engineering Process

Problems with data before re-engineering:

- Data naming problems (file/attributes)

Data Re-engineering

Software Re-engineering Process

Problems with data before re-engineering:

- Field length problems

Data Re-engineering

Software Re-engineering Process

Problems with data before re-engineering:

- Record organization problems

Data Re-engineering

Software Re-engineering Process

Problems with data before re-engineering:

- Hard-coded literals/values

Data Re-engineering

Software Re-engineering Process

Problems with data before re-engineering:

- No data dictionary

Data Re-engineering

Software Re-engineering Process

Problems with data value inconsistencies before re-engineering:

- Inconsistent default value

Different program assign different default value to the same logical data item.

Data Re-engineering

Software Re-engineering Process

Problems with data value inconsistencies before re-engineering:

- Inconsistent validation rules

Data written by one program may be rejected by another

Data Re-engineering

Software Re-engineering Process

Problems with data value inconsistencies before re-engineering:

- Inconsistent units

Data Re-engineering

Software Re-engineering Process

Problems with data value inconsistencies before re-engineering:

- Inconsistent representation semantic

Multiple ways of using uppercase to convey meaning in text strings

Data Re-engineering

Software Re-engineering Process

Problems with data value inconsistencies before re-engineering:

- Inconsistent handling of negative values

Some programs do not accept –ve value, some programs accept & convert to +ve value.

Data Re-engineering

Cost Factors of Re-engineering

- the quality of the software to be re-engineered
- the tool support available for re-engineering
- the extent of data conversion required
- the availability of expert staff

Advantages of Reengineering

- Advantages of re-engineering over developing software from scratch are:
  - reduced risk
  - reduced cost
  - reduced effort
  - reduced development time

Disadvantages of Reengineering

- Practical limits
- Tools support
- Knowledge engineers

Software Re-engineering = Reverse Engineering?

Software Re-engineering = Reverse Engineering?

- NO!
- The main objectives of system re-engineering are to improve the system structure, create new system documentations and make it easier to understand. The main methods of s/w re-engineering :
- i) source code translation, ii) program restructuring, iii) data re-engineering and iv) reverse engineering
- However, the main objective of reverse engineering is to derive system's design and specification from its source code. It is normally part of s/w re-engineering process.

Legacy System Management

- Scrap the system
- Leave it unchanged
- Re-engineer
- Replace all/part

Legacy system management

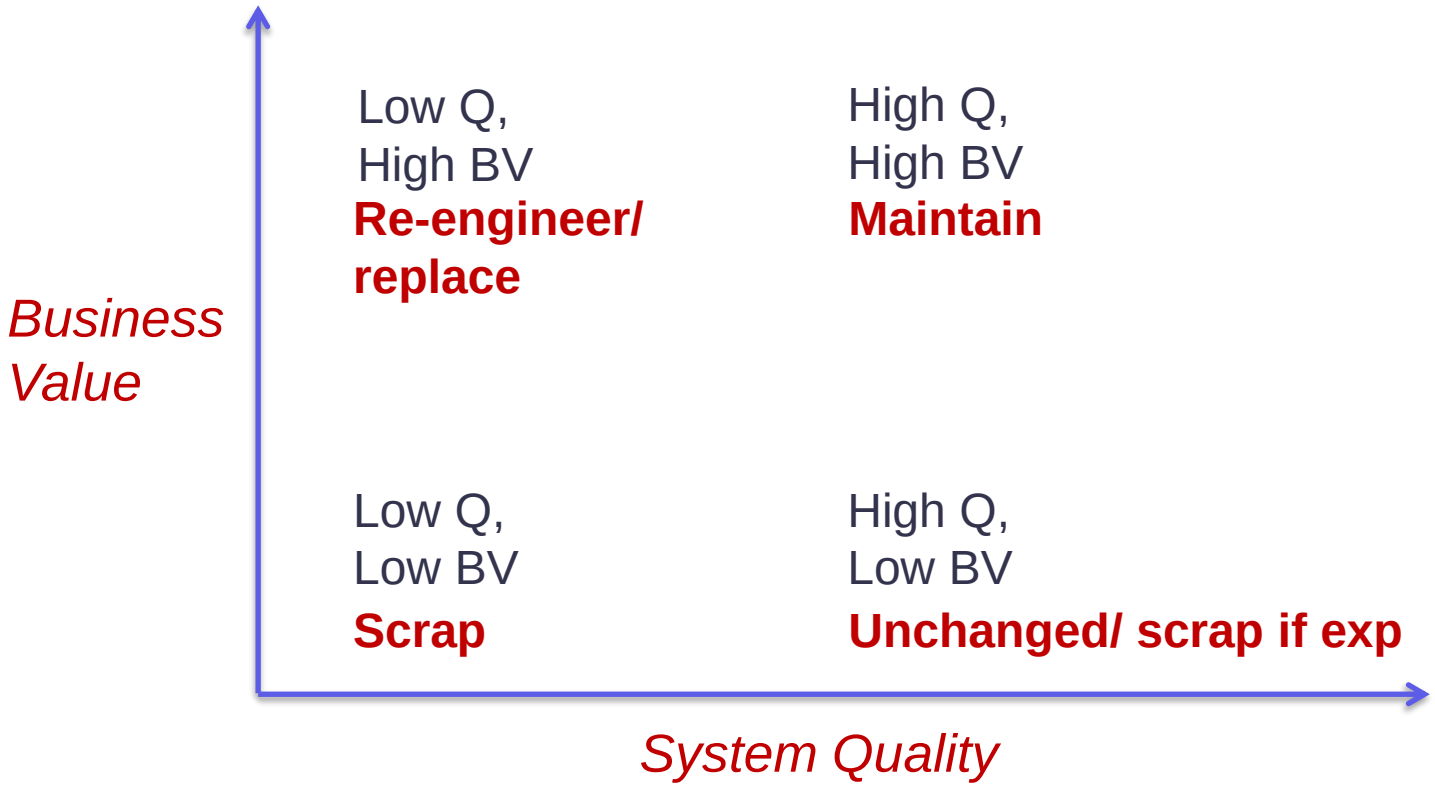
- Many legacy systems are critical business systems
- Four strategies for evolving legacy system:
  - Scrap the system completely
  - Leave the system unchanged and continue with regular maintenance
  - Re-engineer the system to improve it maintainability
  - Replace all or part of the system with a new system

Legacy system management

- When assessing legacy system, both business and technical perspective should consider (Warren, 1998)
  - Low quality, low business value → scrapped
  - Low quality, high business value → re-engineer or replace with suitable off-the-shelf system
  - High quality, low business value → normal system maintenance and scrapped if change become expensive
  - High quality, high business value →Normal system maintenance

Legacy System Management

Legacy system assessment



Exercise

An online trading legacy system in organization ACE serves millions of customer since 1980s. The system maintenance cost is high due to incomplete documentation, poor structured coding and outdated programming language used. Suggest a strategy to the organization to evolve the system.