

ELEC 876: Software Reengineering (Reengineering Economics)

Dr. Ying Zou

Department of Electrical & Computer Engineering
Queen's University



Reengineering Economics

- Measuring software maintenance costs
- Estimating software maintenance effort
- Selecting programs for reengineering
- Estimating the reengineering costs
- Assessing the benefits of reengineering

Typical Questions for Determining the Condition of the Current System

- Does the original system's design and implementation merit reuse in a future system?
- Are new techniques or methodologies exploitable that would improve satisfaction of the system requirements over the original application system?
- Is the target environment vastly different from the original environment and if so how?
- Is redesign of the data necessary? If so, what parts of the process will be impacted?

Typical Questions for Determining the Condition of the Current System (con't)

- What parts of the system require redesign for operation in the target environment?
- Is the system well structured?
- Is the system well maintained? Is the current performance acceptable?
- Is the documentation current?

Calculating Software Maintenance Effort

- COCOMO equation for predicting maintenance effort

$$\text{ANNUAL MAINT-EFFORT} = 1.2 ((\text{ACT}) (\text{DEV-EFFORT})) (1.5 = \text{QUAL})$$

ACT (ANNUAL CHANGE RATE) = Percent of Software deleted, altered and inserted

DEV-EFFORT = Man Months of Development Effort

QUALITY = Quality Factor 0.00 : 1.00

When ACT = 15 %

and DEV-EFFORT = 125 MM

and : ----- QUAL = 0.3 BEFORE REENGINEERING

$$\begin{aligned} \text{ANNUAL MAINT-EFFORT} &= 1.2 ((0.15) (125)) (1.5 - 0.3) \\ &= 27 \text{ Man Months} \end{aligned}$$

IF QUAL = 0.6 AFTER REENGINEERING

$$\begin{aligned} \text{ANNUAL MAINT-EFFORT} &= 1.2 (0.15) (125) (1.5 - 0.6) \\ &= 20 \text{ Man Months} \\ \text{SAVINGS} &= 7 \text{ Man Months per annum} \end{aligned}$$

Software Quality Factors

- Quality is measured by a number of metrics used to calculate a Software Maturity Index (SMI) taken as a quality factor
- IEEE Std. 982.1-1988:

$$SMI = \frac{(M_T - (F_a + F_c + F_d))}{M_T}$$

Where

- M_T : the number of modules in the current release
- F_c : the number of modules in the current release that have been changed
- F_a : the number of modules in the current release that have been added
- F_d : the number of modules in preceding release that have been deleted in the current release

Software Quality Factors (con't)

- Other factors include:
 - Correctness: usually in defects per KLOC
 - Maintainability: usually in Mean Time to Change or cost to correct defects after release spoilage
 - Cohesion: How closely are the parts of a component related.
 - Usually expressed as a real number between 0 (low cohesion) and 1 (high cohesion)
 - Coupling: How independent is a component.
 - Usually represented as a real number between 0 (high coupling) and 1 (low coupling)

Estimating Software Maintenance Costs

- Total maintenance costs without renovation:

$$T_1 = t \cdot (C + P)$$

Where

- T_1 : total maintenance costs without renovation
- t : elapsed time in months
- C : computer costs per month
- P : personnel costs per month

Estimating Software Maintenance Costs (con't)

- Total maintenance cost with renovation:

$$T_1 = t \cdot \left(C + \frac{P}{p} \right) + (N_r \cdot A \cdot t)$$

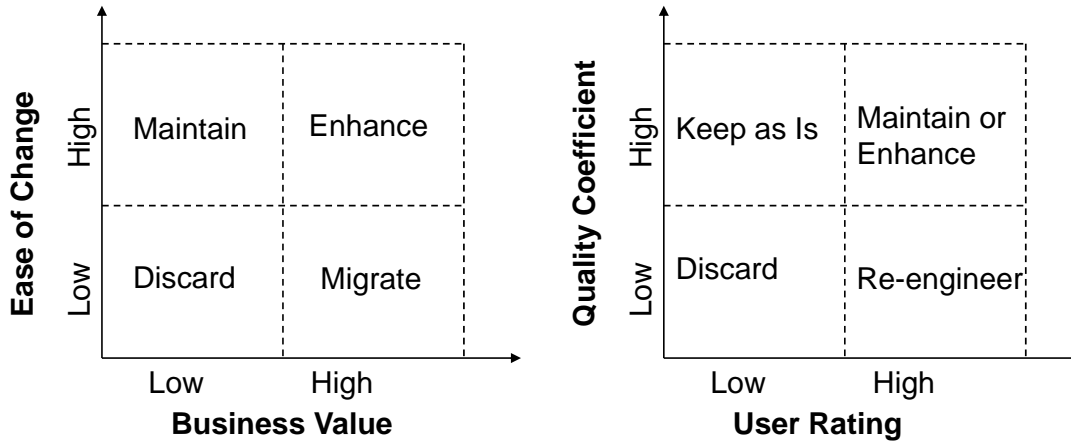
Where

- p: personnel reduction factor as a result of improved maintainability
- N_r : number of people required for the renovation
- A: average monthly cost per person

Selecting Programs for Reengineering

- When the system has a strategic importance
- When the system has not reached its end-of-life-cycle
- When frequent system failures occur
- When the code is over seven years old
- When the program structure and logic flow have become very complex
- When the programs were written for a previous generation hardware
- When the programs are running in emulation mode
- When the modules or unit subroutines have grown excessively large
- When excessive resources are required to run the system
- When hard-coded parameters are subject to change
- When it becomes difficult to retain the maintainers
- When the documentation has become out of date
- When the design specifications are missing, incomplete or obsolete

Selecting Criteria for Reengineering



Estimating Reengineering Effort

- Empirical reengineering efforts per type of application

Component	Manual	Automatic
Program (2000 LOC)	5	1 PD
Subroutine (100 LOC)	2	0.5 PD
Job	4	2 PD
Utility	2	0.5 PD
Database	8	4 PD
File	4	2 PD
Copy/include	1	0.5 PD
Parameter	1	1 PD
Panel	1	0.5 PD

28

12 PD

50% cost reduction through automation

Estimating Reengineering Effort (con't)

- Reengineering effort calculation

$$\begin{aligned}
 E_1 &= (\text{Number of programs} \times \text{Program effort}) \times \\
 &\quad (1 + (\text{Number IOs/Number of programs} + \text{Number IOs})) \\
 E_2 &= (\text{Number of subroutines} \times \text{Subroutine effort}) \times \\
 &\quad (1 + (\text{Number CALLs/Number of subroutines} + \text{Number CALLs})) \\
 E_3 &= (\text{Number jobs} \times \text{Job effort}) \times \\
 &\quad (1 + (\text{Number files/Number jobs} + \text{Number files})) \\
 E_4 &= (\text{Number files} \times \text{File effort}) \times \\
 &\quad (1 + (\text{Number IOs/Number files} + \text{Number IOs})) \\
 E_5 &= (\text{Number copys} \times \text{Copy effort}) \times \\
 &\quad (1 + (\text{Number ref/Number copys} + \text{Number ref})) \\
 E_6 &= (\text{Number panels} \times \text{Panel effort}) \times \\
 &\quad (1 + (\text{Number fields/Number panels} + \text{Number fields})) \\
 E &= (E_1 + E_2 + E_3 + E_4 + E_5 + E_6) \times 2 \text{ [Test effort]} \\
 E &= \text{Total re-engineering effort}
 \end{aligned}$$

Sample Estimation of the Reengineering Effort

- Sample estimation of the reengineering effort in Person Days (PD)

Sample-system
 500 Programs with 2000 accesses
 100 Subroutines with 400 CALLs
 50 Jobs with 200 files
 200 Files with 2000 accesses
 300 COPYs with 3000 references
 200 Panels with 2000 fields

$$\begin{aligned}
 E_1 &= (500 \times 1) \times (1 + 2000/2500) &= 900 \text{ PD} \\
 E_2 &= (100 \times 0.5) \times (1 + 400/500) &= 90 \text{ PD} \\
 E_3 &= (50 \times 2) \times (1 + 200/250) &= 180 \text{ PD} \\
 E_4 &= (200 \times 2) \times (1 + 2500/2700) &= 760 \text{ PD} \\
 E_5 &= (300 \times 0.5) \times (1 + 3000/3300) &= 285 \text{ PD} \\
 E_6 &= (200 \times 0.5) \times (1 + 2000/2200) &= 190 \text{ PD}
 \end{aligned}$$

2405 PD
 × 2

PD = Person-days
 PM = Person-months

4810 PD
 ↓
 240 PM

Assessing the Benefits of Reengineering

- Three main cases:

- Application is technically obsolete and must be replaced

$$R_{benefit} = (V_{old} - (R_{cost} \cdot R_{risk})) - (V_{new} - (Dev_{cost} \cdot Dev_{risk}))$$

- Application has serious technical problems that lead to increased maintenance costs

$$R_{benefit} = (M_{old} - M_{new}) + (V_{old} - (R_{cost} \cdot R_{risk})) - (V_{new} - (Dev_{cost} \cdot Dev_{risk}))$$

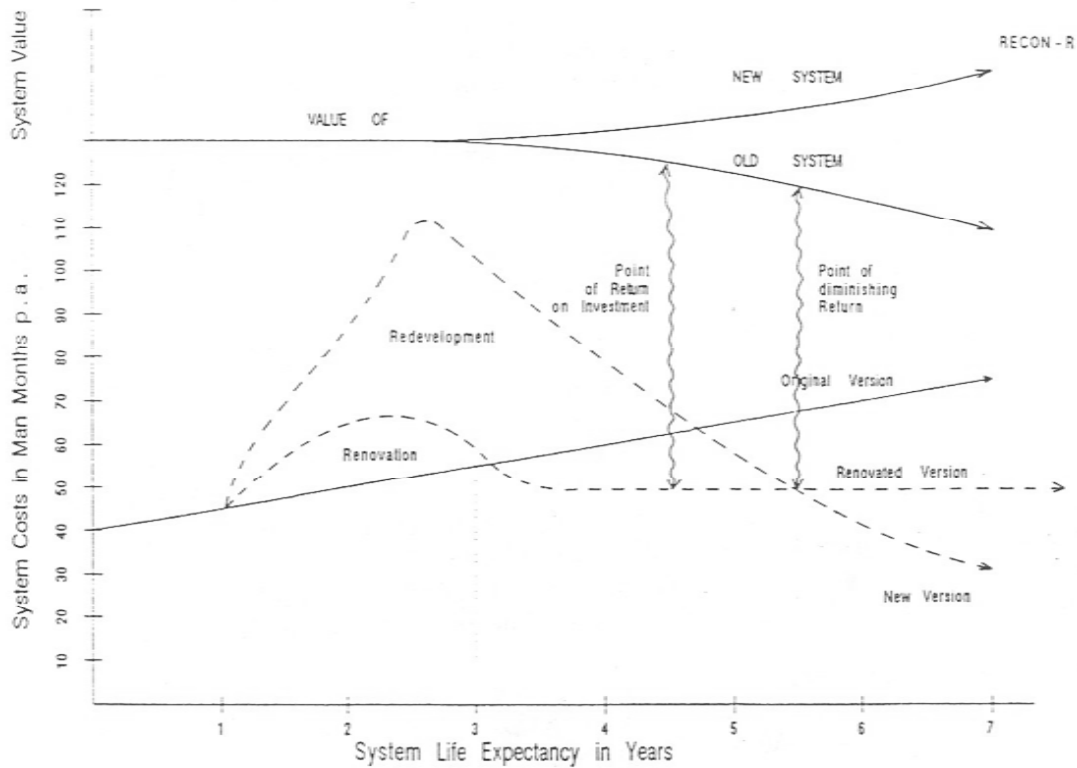
- Application has relatively higher than expected maintenance costs

$$R_{benefit} = 2 \cdot M_{acr} - (R_{cost} \cdot R_{risk})$$

Assessing the Benefits of Reengineering (con't)

Where

- $R_{benefit}$: the reengineering benefit (must be positive to decide to reengineer)
- V_{old} , V_{new} : the value of the old and the new system
- M_{old} , M_{new} , M_{acr} : the maintenance cost of the old system, the new system, and the annual maintenance cost reduction due to reengineering, respectively
- R_{cost} , R_{risk} : reengineering cost and reengineering risk (3 high, 1 low), respectively
- Dev_{cost} , Dev_{risk} : development cost and development risk, respectively



Fall 2007

Elec 876: Software Reengineering

17

State of Practice on Assessing Reengineering Efforts

- There are no accurate success/failure measures for reengineering projects
- Success criteria must be agreed at the beginning of the project
- Measurement systems are needed to provide a comparison between
 - Source and target system's functionality, efficiency, maintainability in accordance with other factors such as cost, effort, system value
 - Code redundancy (before and after reengineering)
- Technical vs. business success criteria
- Reuse as a focal point

Fall 2007

Elec 876: Software Reengineering

18

State of Practice on Assessing Reengineering Efforts (con't)

- Is better to analyze code than high level designs
- The reengineering project should aim for
 - A more usable system
 - A more adaptable system