CHAPTER 7

SOFTWARE MAINTENANCE AND RE-ENGINEERING

Neither situation nor people can be altered by the interference of an outsider. If they are to be altered, that alteration must come from within.

Phyllis Bottome

SOFTWARE MAINTENANCE

- It begins immediately when software is released to end users, and bug reports back to the software engineering organization.
- The challenge of software maintenance are queue of bug fixes, adaptation requests and enhancements that must be planned, scheduled, and ultimately accomplished.
- Organization finds that it's spending more money and time maintaining existing programs than it is engineering new applications.
- It is not unusual for a software organization to expend as much as 60 to 70 percent of all resources on software maintenance.

SOFTWARE MAINTENANCE CON'T …

- Even programs created using the best design and coding techniques known at the time ,they were then migrated to new platforms, adapted to Hardware & OS technology and enhanced to meet new user needs—all without enough regard to overall architecture.
- The result is the **poorly designed structures**, **coding**, **logic**, **and documentation** of the software systems
- *Maintainability* is a qualitative indication of existing software can be corrected, adapted, or enhanced.
- Hence software Engineering is building systems that exhibit high maintainability.

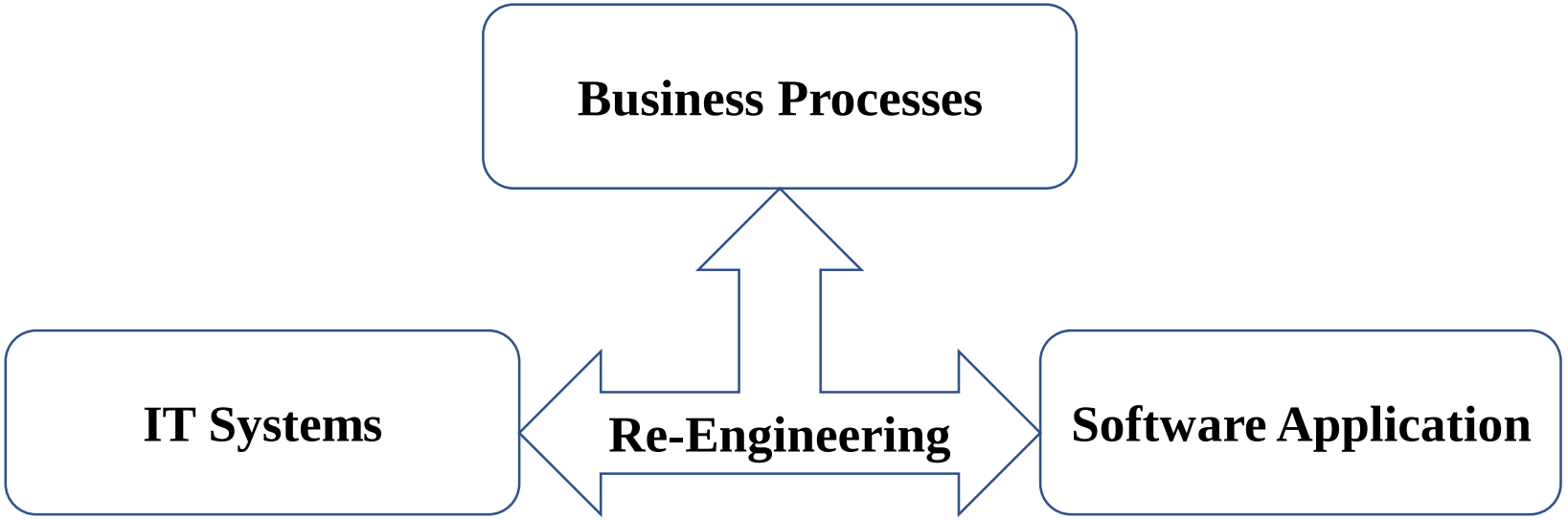
MAINTAINABILITY

- Maintainable software exhibits effective modularity .
- It makes use of design patterns that allow ease of understanding.
- It has been constructed using well-defined coding standards and conventions, leading to source code that is self-documenting and understandable.
- It has undergone a variety of quality assurance techniques that uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not been around when changes must be made.
  - Therefore, the design and implementation of the software must "assist" the person who make a changes.

SOFTWARE SUPPORTABILITY

- The capability of supporting a software system over its whole product life.
  - This implies satisfying any necessary needs or requirements, but also the provision of equipment, support infrastructure, additional software, facilities, manpower, or any other resource required to maintain the software operational and capable of satisfying its function.
- The software should contain facilities to assist support personnel when a defect is encountered in the operational. In addition, support personnel should have access to a database that contains records of all defects that have already been encountered—their characteristics, cause, and cure.

REENGINEERING



- When Use the power of modern information technology to radically redesign our business processes in order to achieve dramatic improvements in their performance.

REENGINEERING CON'T

- A **business process** is "a set of logically related tasks performed to achieve a defined business outcome".
- Within the business process, people, equipment, material resources, and business procedures are combined to produce a specified result.
- Examples of business processes include designing a new product, purchasing services and supplies, hiring a new employee, and paying suppliers.
- Each demands a set of tasks, and each draws on diverse resources within the business.

A BPR MODEL CON'T…

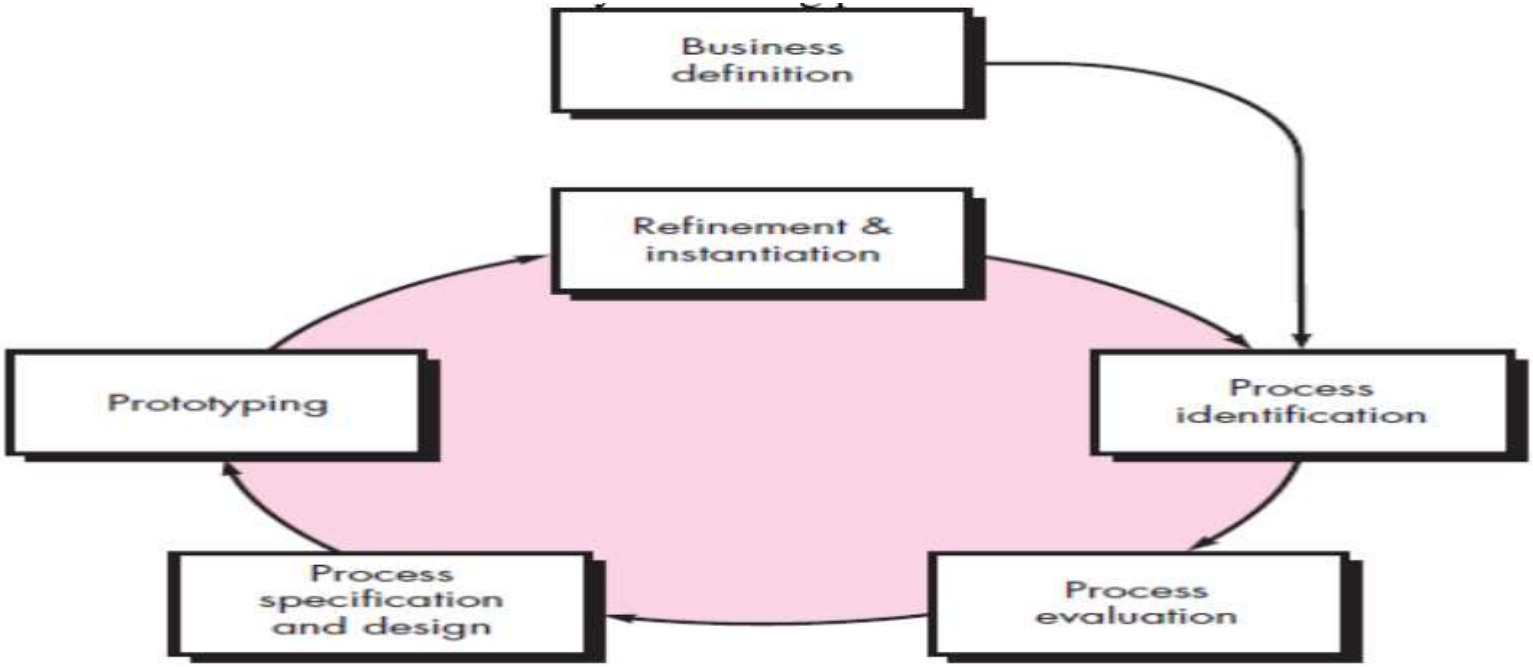
- Like most engineering activities, business process reengineering is iterative.
- Business goals and the processes that achieve them must be adapted to a changing business environment.
- There is no start and end to BPR—it is an evolutionary process.
- The model defines six activities:

i. **Business definition.** Business goals are identified within the context of four key drivers: *cost reduction*, *time reduction*, *quality improvement*, and *personnel development and empowerment*. Goals may be defined at the business level or for a specific component of the business.

A BPR MODEL

ii. **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified.

- They may then be ranked by importance, by need for change, or in any other way that is appropriate for the reengineering activity.



A model for business process reengineering is depicted in Figure.

A BPR MODEL CON'T…

iii. **Process evaluation.** The existing process is thoroughly analyzed and measured. Process tasks are identified; the costs and time consumed by process tasks are noted; and quality/performance problems are isolated.

iv. **Process specification and design.** Based on information obtained during the first three BPR activities, use cases are prepared for each process that is to be redesigned. With the use case as the specification of the process, a new set of tasks are designed for the process.

v. **Prototyping.** A redesigned business process must be prototyped before it is fully integrated into the business. This activity "tests" the process so that refinements can be made.

A BPR MODEL CON'T …

vi. **Refinement and instantiation.** Based on feedback from the prototype, the business process is refined and then instantiated within a business system.

- These BPR activities are sometimes used in conjunction with workflow analysis tools. The intent of these tools is to build a model of existing workflow in an effort to better analyze existing processes.

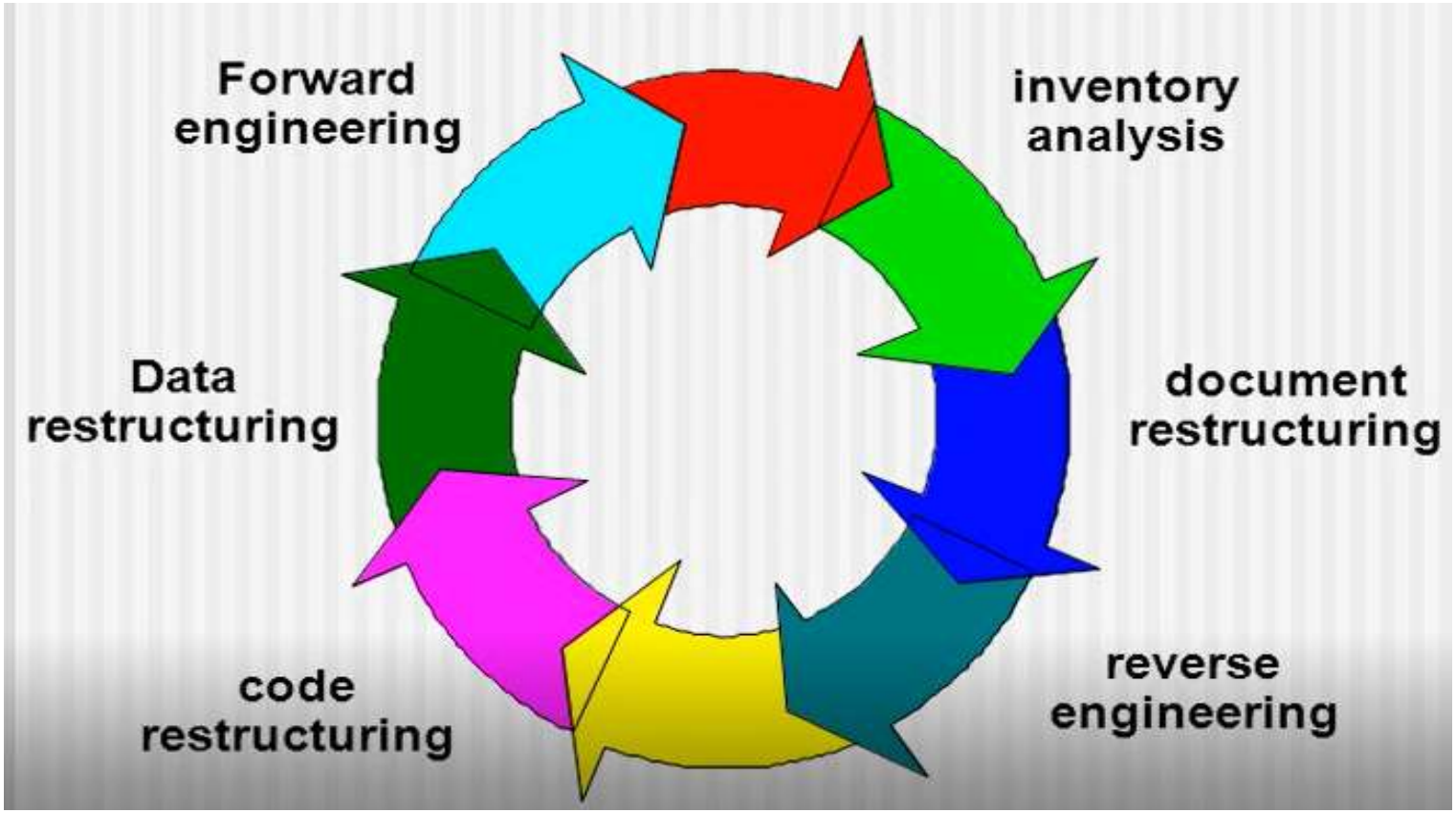
SOFTWARE REENGINEERING

- Reengineering takes time, it costs significant amounts of money, and it absorbs resources.
- Reengineering of information systems is an activity that will absorb information technology resources for many years. Reengineering is a rebuilding activity.

SOFTWARE REENGINEERING

Software Reengineering Activities

- The reengineering paradigm shown in Figure is a cyclical model. This means that each of the activities presented as a part of the paradigm may be revisited. For any particular cycle, the process can terminate after any one of these activities.



SOFTWARE REENGINEERING

I. Inventory analysis:

- Build a table that contains all applications
- Establish a list of criteria, e.g.
  - Name of the application
  - Year it was originally created
  - Number of substantive changes made to it
  - Total effort applied to make these changes
  - Date of last substantive changes
  - Effort applied to make the last change
  - Systems in which it resides
  - Applications to which it interfaces, …
- Analyze and prioritize to select candidates for reengineering

SOFTWARE REENGINEERING

II. Document restructuring:

- Weak documentation is the trademark of many legacy systems. But what can you do about it? What are your options?

Options…

1. *Creating documentation is far too time consuming*.

- If the system works, you may choose to live with what you have. In some cases, this is the correct approach.

2. *Documentation must be updated, but your organization has limited resources.*

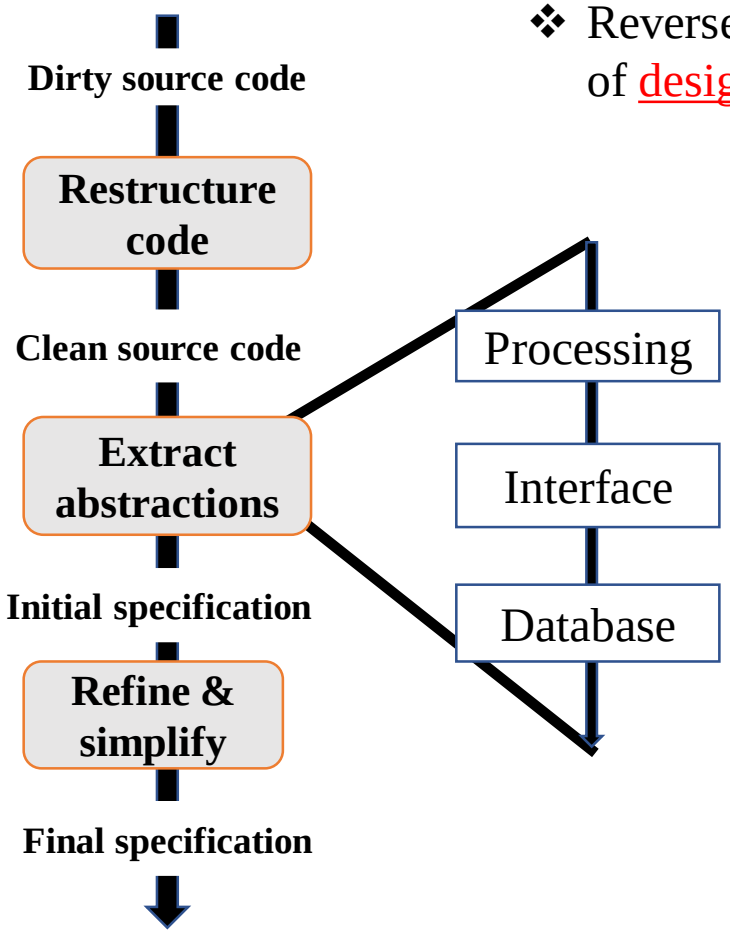
- You'll use a "document when touched" approach. It may not be necessary to fully redocument an application.

3. *The system is business critical and must be fully redocumented*.

- Even in this case, an intelligent approach is to pare documentation to an essential minimum.

SOFTWARE REENGINEERING

III. Reverse engineering: Reverse engineering for software is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code.

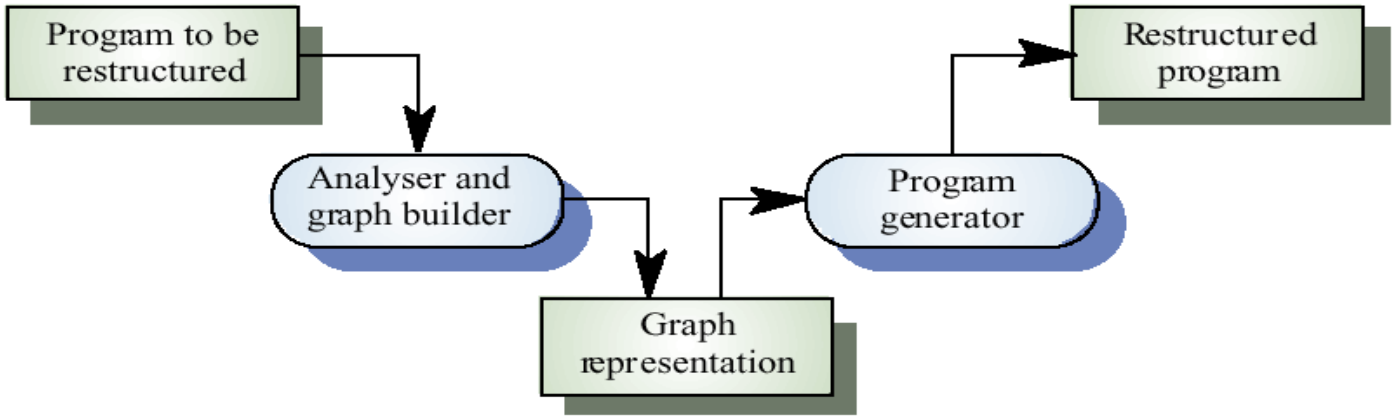


- Reverse engineering is a process of design recovery.

SOFTWARE REENGINEERING

IV. Code restructuring. Some legacy systems have a relatively solid program architecture, but individual modules were coded in a way that makes them difficult to understand, test, and maintain.

- In such cases, the code within the suspect modules can be restructured.
- The source code is analyzed using a restructuring tool.
- Violations of structured programming constructs are noted and code is then restructured (this can be done automatically).
- The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced. Internal code documentation is updated.



SOFTWARE REENGINEERING

V. Data restructuring:

- for many applications, information architecture has more to do with the long-term viability of a program than the source code itself.
- Unlike code restructuring, which occurs at a relatively low level of abstraction, data restructuring is a full-scale reengineering activity.
- In most cases, data restructuring begins with a reverse engineering activity.
  - Current data architecture is dissected and necessary data models are defined.
  - Data objects and attributes are identified, and existing data structures are reviewed for quality.
  - When data structure is weak the data are reengineered.
- Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

SOFTWARE REENGINEERING

VI. Forward engineering.

1. The cost to maintain one line of source code may be 20 to 40 times the cost of initial development of that line.
2. Redesign of the software architecture (program and/or data structure), using modern design concepts, can greatly facilitate future maintenance.
3. Because a prototype of the software already exists, development productivity should be much higher than average.
4. The user now has experience with the software. Therefore, new requirements and the direction of change can be ascertained with greater ease.
5. Automated tools for reengineering will facilitate some parts of the job.
6. A complete software configuration (documents, programs, and data) will exist upon completion of preventive maintenance.

- The forward engineering process applies software engineering principles, concepts, and methods to re-create an existing application.

THE ECONOMICS OF REENGINEERING

- A cost-benefit analysis model for reengineering has been proposed by Sneed [Sne95]. Nine parameters are defined:

  *P*1 _ current annual maintenance cost for an application
  *P*2 _ current annual operations cost for an application
  *P*3 _ current annual business value of an application
  *P*4 _ predicted annual maintenance cost after reengineering
  *P*5 _ predicted annual operations cost after reengineering
  *P*6 _ predicted annual business value after reengineering
  *P*7 _ estimated reengineering costs
  *P*8 _ estimated reengineering calendar time
  *P*9 _ reengineering risk factor (*P*9 _ 1.0 is nominal)
  L _ expected life of the system

THE ECONOMICS OF REENGINEERING

The cost associated with continuing maintenance of a candidate application (i.e., reengineering is not performed) can be defined as

$$C_{\text{maint}} = [P_3 - (P_1 + P_2)] \times \text{L} \tag{29.1}$$

The costs associated with reengineering are defined using the following relationship:

$$C_{\text{reeng}} = P_6 - (P_4 + P_5) \times (L - P_8) - (P_7 \times P_9) \tag{29.2}$$

Using the costs presented in Equations (29.1) and (29.2), the overall benefit of reengineering can be computed as

$$\text{Cost benefit} = C_{\text{reeng}} - C_{\text{maint}} \tag{29.3}$$

The cost-benefit analysis presented in these equations can be performed for all high-priority applications identified during inventory analysis (Section 29.4.2). Those applications that show the highest cost-benefit can be targeted for reengineering, while work on others can be postponed until resources are available.

REFERENCE

1. Software engineering: A practitioner’s approach, roger s. preeman, (chapter 29)