REENGINEERING
Table of Contents

1. Introduction/Quick Review
2. Business Process Re-engineering
   2.1 BPR Model
3. Software Re-engineering
   3.1 Process Model
   3.2 Stages of Re-engineering
4. Reverse Engineering
   4.1 The reverse engineering process
   4.2 Reverse Engineering to Understand Processing
   4.3 Reverse Engineering to Understand Data
   4.4 Reverse Engineering User Interfaces
# Table of Contents

5. Restructuring  
   5.1 Problems of Restructuring  
   5.2 Code Restructuring  
   5.3 Data Restructuring  
6. Forward Engineering
1. Introduction/Quick Review

“...we should reengineer our businesses: use the power of modern information technologies to radically redesign our business processes in order to achieve dramatic improvements in their performance...Reengineering strives to break away from the old rules about how we organize and conduct our business...”

Michael Hammer, in Harvard Business Review
1. Introduction/Quick Review

- What is it?:
  - a technology product has served well, you rely on it but its getting old and breaks to often
  - it takes longer to repair than you like
  - it doesn’t longer represent the state-of-art technology

- What to do?
  - hardware → throw away, buy new
  - custom-built software → need for rebuild
    - create a product with added functionality, better performance, reliability and improved maintainability
    - that is what we call re-engineering
1. Introduction/Quick Review

- Who is executing?
  - at business level → business specialists (consulting companies)
  - at the software level → software engineers

- Why is it important?
  - IT supports fast changing business functions on demand that causes enormous pressure on every commercial organisation
    - both business and software that support the business must be reengineered to keep pace
1. Introduction/Quick Review

- What are the steps of Business-Process-Reengineering (BRP)?
  - define goals
  - identifying and evaluating existing processes
  - create new processes that better meet the goals

- What are the steps of Software-Reengineering?
  - encompassing inventory analysis
  - document restructuring
  - program and data reconstruction
  - forward engineering
  - **Intent:** creation of a version existing program with higher quality and better maintainability
1. Introduction/Quick Review

- What is the work product?:
  - reengineering products like:
    - analysis models
    - design models
    - test procedures
  - final output: reengineered and software that support the reengineered process
- Have I done it right?:
  - use the well known SQA practices like
    - formal reviews
    - assessment of analysis and design models
    - testing for errors, functionality and interoperability
2. Business Process Reengineering

- **Definition**
  - no unique definition
  - “… the search for, and the implementation of, radical change in business process to achieve breakthrough results…” (Fortune Magazine)

- **Business Processes**
  - set of logical related tasks for defined output
  - combining people, equipment, material, procedures
  - e.g. designing new product, purchasing services and supplies, hiring a new employee, paying suppliers…
  - receiver of the outcome: the customer
  - cross organizational boundaries, lots of organizational groups participating
2. Business Process Reengineering

- business segments:
  - the business
  - business systems
  - business process
  - business sub processes

- most BPR efforts focus on individual processes or sub processes
2. Business Process Reengineering

- Principles of Business Process Reengineering
  - top down manner
    - beginning with the identification of major business objectives and goals and culminating with more detailed specification of tasks
  - principals that guide BPR activities:
    - organize around outcomes, not tasks
    - let users of the output of the process perform the process
    - integrate information processing work into real work
    - treat geographical disperse resources as them were centralized
    - link parallel activities instead of integrating their results
    - set decision points, where the work is performed, process based control
    - capture data once, at its source
2.1 BPR Model

- Business definition
- Refinement & instantiation
- Prototyping
- Process identification
- Process specification and design
- Process evaluation
2.1 BPR Model

- business reengineering iterative process
- no start, no end, evolutinal
- six elements:
  - business definition:
    - identify goals by using four drivers (cost reduction, time reduction, quality improvement, personal development)
    - define at the business level or for a specific component
  - process identification:
    - identify critical processes for goals defined in the business definition
    - rank them by importance, need for change…
  - process evaluation:
    - measure and identify tasks
    - note costs and time consumption
    - isolate quality/performance
2.1 BPR Model

- Process specification and design:
  - prepare use-cases (for each process) based on the information collected during the first three steps
  - identify use-cases to scenarios that deliver outcome to the customers
  - use-cases → design new set of tasks (conform with Software Engineering principles)

- Prototyping:
  - test the process to make refinements

- Refinement and instantiation:
  - based on feedback from the prototype
  - refine and instantiate with the business system

- intent of these tools: building a model of existing workflow in an effort to better analyze existing processes
Warning

- BPR works if it is applied by motivated, trained, people
- reengineering = continuous activity
- if BPR is conducted efficiently \(\rightarrow\) information system better support the business process
- software reengineering something that have to be done because of the needs that a lot of business applications need to be refurbish or rebuild
3. Software Reengineering

- software was used for a long time; corrected, adapted, enhanced many times
- unstable application, side effects etc.
- there is always change in computer science → mechanisms for evaluating, controlling and making modifications
- Software Maintenance (60% of costs): more than just „fixing mistakes“
- also: Reengineering (Rebuilding an existing Software System)
3.1 Process Model

- reengineering needs time (maybe years) and money → pragmatic strategy

- mostly linear
- cyclical model

Source: Software Engineering, Roger S. Pressman, Page 807
3.2 Stages of Reengineering

- **Inventory Analysis**
  - inventory: attributes of application (size, age, business criticality, etc.)
  - changes → revisit inventory regularly!

- **Document Restructuring**
  - no re-documentation (relatively static, not much longer used, no changes expected)
  - update documentation when changes were made
  - complete re-documentation, but kept at a minimum (business critical systems)
3.2 Stages of Reengineering

- Reverse Engineering
  - analyze program code to create a representation of the program at a higher level of abstraction (e.g. UML diagrams)
  - design and specification recovery

- Code Restructuring
  - most common type
  - analyze source code (restructuring tool), possible automatically restructuring
  - review and test the result, update code documentation!
3.2 Stages of Reengineering

- Data Restructuring
  - restructuring begins mostly with reverse engineering (define/review data models and structures)
  - restructure weak data architectures
  - data restructuring requires changes in architectural or source code changes

- Forward Engineering
  - recover design information from existing software AND alter/restore system → improve overall quality
  - re implement old function, add new ones, improve overall performance and quality
4. Reverse Engineering

- possibly part of the reengineering process
- can used to re-specify a system for re-implementation
- used to analyze software and get a better comprehension about design and specification
- creates a program database and generates information from this
- use of tools that understand the program is helpful
  - browsers
  - cross-reference generators
4.1 The reverse engineering process

System to be re-engineered

Automated analysis

Manual annotation

System information store

Document generation

Program structure diagrams

Data structure diagrams

Traceability matrices
4.2 Reverse Engineering to Understand Processing

- analyzing code at varying levels of abstraction (system, program, component, pattern, statement)
- important → interoperability issues among application within the system
- block diagram, representing the interactions between functional abstractions of all programs of the system
- processing narrative for each component of a program
- update of specifications (if system, program, component specification already exists)
- CASE Tools to analyze semantic of existing code automatically → Restructuring / Forward Engineering
4.3 Reverse Engineering to Understand Data

- Internal Data Structures (Program Level)
  - identifying classes within the program that interact with the global data structures
  - examining program code with intent of grouping related program variables

Three Steps for reverse engineering of classes
- identify flags and local data structures, that holds important information about global data structures
- define relationship between flags and local data structures and global data structures
- list all other variables that are connected to Array/File structures
4.3 Reverse Engineering to Understand Data

- Database Structure (System Level)
  - Reengineering one database schema into another requires knowledge about existing objects / relationships

Five Steps to reengineer a new database model

- Build initial object Model
- Determine candidate keys
- Refine the temporary classes
- Define generalizations
- Discover associations
4.4 Reverse Engineering User Interfaces

- What are the basic actions that the Interface must process (e.g., keystrokes, mouse clicks)?
- What is a compact description of the behavioral response of the system to these actions?
- What concept equivalence of interface is relevant here?

Result:
- revised GUI may not mirror the old interface
- often contains new interaction metaphors
5. Restructuring

- modifying source code/data to make it accessible for further changes
- no change of the overall program architecture
- effort extends beyond module boundaries, encompasses software architecture → Forward Engineering

Beneficial effects:
- higher quality of programs, better documentation
- compliance to modern software engineering practices and standards
- structured code facilitate co-operation among involved software developers
- less maintenance cost
- software easier to test and debug
5.1 Problems of Restructuring

- Loss of Information
  - Comments
  - Documentation
  - Heavy computational demands
- not useful help with poor modularisation where related components are dispersed throughout the code
- not to improve the understand ability of data-driven programs.
5.2 Code Restructuring

- to reach a design that is congruent with former functions, but higher quality than original program
- using Boolean algebra for applying a series of transformation rules
- goals:
  - reach restructured logic
  - derive a procedural design that conforms the structured programming philosophy from former “spaghetti-bowl” code
### Spaghetti Code → Structured Code

#### Code

```c
Start: Get (Time-on, Time-off, Time, Setting, Temp, Switch)
      if Switch = off goto off
      if Switch = on goto on
      goto Cntrlid
off: if Heating-status = on goto Sw-off
      goto loop
on: if Heating-status = off goto Sw-on
      goto loop
Cntrlid: 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
```

---

```c
loop
-- The Get statement finds values for the given variables from the system's
-- environment.
Get (Time-on, Time-off, Time, Setting, Temp, Switch) ;
case Switch of
    when On => if Heating-status = off then
        Switch-heating; Heating-status := on ;
        end if ;
    when Off => if Heating-status = on then
        Switch-heating; Heating-status := off ;
        end if ;
    when Controlled =>
        if Time >= Time-on and Time <= Time-off then
            if Temp > Setting and Heating-status = on then
                Switch-heating; Heating-status := off ;
                end if ;
            elseif Temp <= Setting and Heating-status = off then
                Switch-heating; Heating-status := on ;
                end if ;
            end if ;
        end if ;
end case ;
end loop ;
```
5.3 Data Restructuring

1. data analysis (analysis of source code)
   - evaluation of all program language statements that contains data definitions, I/O and Interface descriptions
   - extract data items and objects to get information about data flow and comprehend implemented data structures

2. data redesign
   - data record standardization → simplest form
     (clarify data definitions to reach consistency concerning data item names or physical record formats within an existing data structure)
   - data name rationalization
     (used data naming conventions must conform to local standards and aliases have to be eliminated as data flow through the system)
6. Forward Engineering

- software engineering principles, concepts and methods to re-create an existing application
- mostly not a modern equivalent of older program → rather new user and technology requirements are integrated
- re-created program has more functionality than older program
- preventive maintenance → „new releases“ of a program
6. Forward Engineering

- consider these points:
  - can make future maintenance easier when using modern design concepts
  - development productivity is much higher than average (software exists)
  - new requirements and direction of change can be defined with greater ease (user knows software)
  - CASE tools for reengineering → automate some parts of the job
  - preventive maintenance done → complete software configuration will exist