Course : COMP6100/Software Engineering

Effective Period : Desember 2017

Risk Analysis, Maintenance and Reengineering

Session 31

People
Innovation
Excellence

BINUS UNIVERSITY

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 35 and 36

People
Innovation
Excellence

Learning Objectives

LO 4 : Analyze the software project management and the proposed potential business project

Contents

- Reactive versus Proactive Risk Strategies
- Software Risk
- Software Maintenance
- Business Process Reengineering
- Software Reengineering
- Reverse Engineering
- Forward Engineering

People
Innovation
Excellence

Reactive versus Proactive Risk Strategies

Reactive Risk Management

- project team reacts to risks when they occur
- mitigation—plan for additional resources in anticipation of fire fighting
- fix on failure—resource are found and applied when the risk strikes
- crisis management—failure does not respond to applied resources and project is in jeopardy

Reactive versus Proactive Risk Strategies

Proactive Risk Management

- formal risk analysis is performed
- organization corrects the root causes of risk
  - TQM concepts and statistical SQA
  - examining risk sources that lie beyond the bounds of the software
  - developing the skill to manage change

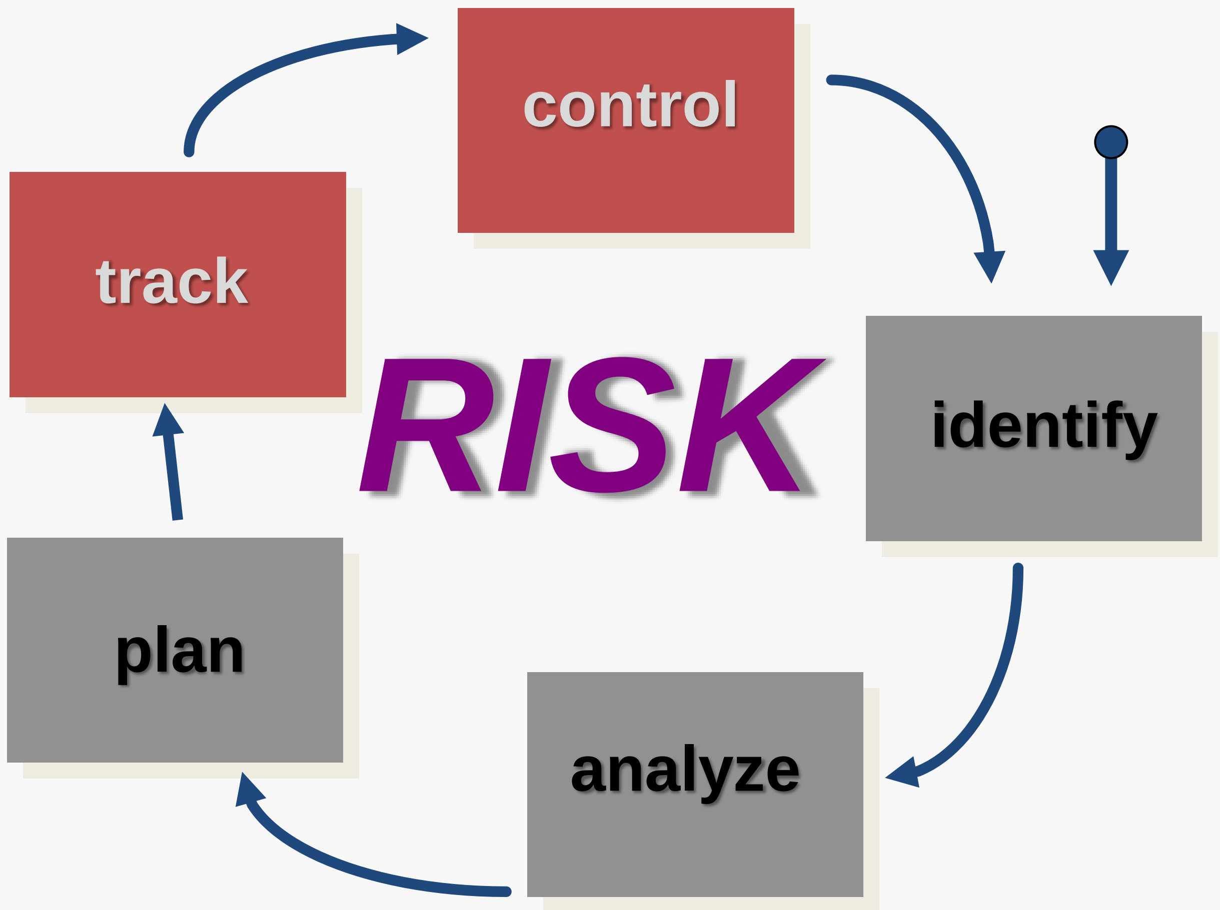
Software Risk

Seven Principles

- **Maintain a global perspective**—view software risks within the context of system and the business problem
- **Take a forward-looking view**—think about the risks that may arise in the future; establish contingency plans
- **Encourage open communication**—if someone states a potential risk, don’t discount it.
- **Integrate**—a consideration of risk must be integrated into the software process
- **Emphasize a continuous process**—the team must be vigilant throughout the software process, modifying identified risks as more information is known and adding new ones as better insight is achieved.
- **Develop a shared product vision**—if all stakeholders share the same vision of the software, it likely that better risk identification and assessment will occur.
- **Encourage teamwork**—the talents, skills and knowledge of all stakeholder should be pooled

Software Risk

Risk Management Paradigm



People
Innovation
Excellence

BINUS UNIVERSITY

Software Risk

Risk Identification

- ***Product size*** —risks associated with the overall size of the software to be built or modified.
- ***Business impact*** —risks associated with constraints imposed by management or the marketplace.
- ***Customer characteristics*** —risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- ***Process definition*** —risks associated with the degree to which the software process has been defined and is followed by the development organization.
- ***Development environment*** —risks associated with the availability and quality of the tools to be used to build the product.
- ***Technology to be built*** —risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.
- ***Staff size and experience*** —risks associated with the overall technical and project experience of the software engineers who will do the work.

Software Risk

Assessing Project Risk-I

- Have top software and customer managers formally committed to support the project?
- Are end-users enthusiastically committed to the project and the system/product to be built?
- Are requirements fully understood by the software engineering team and their customers?
- Have customers been involved fully in the definition of requirements?
- Do end-users have realistic expectations?

Software Risk

Assessing Project Risk-II

- Is project scope stable?
- Does the software engineering team have the right mix of skills?
- Are project requirements stable?
- Does the project team have experience with the technology to be implemented?
- Is the number of people on the project team adequate to do the job?
- Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

People
Innovation
Excellence

Software Risk

Risk Components

- ***performance risk***—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- ***cost risk***—the degree of uncertainty that the project budget will be maintained.
- ***support risk***—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- ***schedule risk***—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

Software Risk

Risk Projection

- *Risk projection*, also called *risk estimation,* attempts to rate each risk in two ways
  - the likelihood or probability that the risk is real
  - the consequences of the problems associated with the risk, should it occur.
- The are four risk projection steps:
  - establish a scale that reflects the perceived likelihood of a risk
  - delineate the consequences of the risk
  - estimate the impact of the risk on the project and the product,
  - note the overall accuracy of the risk projection so that there will be no misunderstandings.

Software Risk

Impact assessment

Category		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values in excess of $500K	
	2	Significant degradation to nonachievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected value of $100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and/or schedule impact with expected value of less than $1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget underrun	Early achievable IOC

Note: (1) The potential consequence of undetected software errors or faults.
(2) The potential consequence if the desired outcome is not achieved.

Software Risk

Building the Risk Table

- Estimate the probability of occurrence
- Estimate the impact on the project on a scale of 1 to 4, where
  - 1 = catastrophic
  - 2 = critical
  - 3 = marginal
  - 4 = negligible
- sort the table by probability and impact

Software Risk

Sample risk table prior to sorting

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover will be high	ST	60%	2	
⋮				

Impact values:
1—catastrophic
2—critical
3—marginal
4—negligible

People
Innovation
Excellence

Software Risk

Risk Exposure (Impact)

The overall *risk exposure,* RE, is determined using the following relationship [Hal98]:

$$\mathbf{RE} = \boldsymbol{P} \times \boldsymbol{C}$$

where

P is the probability of occurrence for a risk, and

C is the cost to the project should the risk occur.

Software Risk

Risk Exposure Example

- **Risk identification.** Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.
- **Risk probability.** 80% (likely).
- **Risk impact.** 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.
- **Risk exposure.** RE = 0.80 x 25,200 ~ $20,200.

Software Risk

Risk Mitigation, Monitoring, and Management

- Mitigation —how can we avoid the risk?
- Monitoring —what factors can we track that will enable us to determine if the risk is becoming more or less likely?
- Management —what contingency plans do we have if the risk becomes a reality?

BINUS UNIVERSITY

Software Risk

Risk information sheet

Risk information sheet			
Risk ID: P02-4-32	Date: 5/9/09	Prob: 80%	Impact: high

Description:
Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

Refinement/context:
Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards.
Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.

Mitigation/monitoring:
1. Contact third party to determine conformance with design standards.
2. Press for interface standards completion; consider component structure when deciding on interface protocol.
3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.

Management/contingency plan/trigger:
RE computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly.
Trigger: Mitigation steps unproductive as of 7/1/09.

Current status:
5/12/09: Mitigation steps initiated.

Originator: D. Gagne	Assigned: B. Laster

People
Innovation
Excellence

Software Maintenance

- Software is released to end-users, and
  - within days, bug reports filter back to the software engineering organization.
  - within weeks, one class of users indicates that the software must be changed so that it can accommodate the special needs of their environment.
  - within months, another corporate group who wanted nothing to do with the software when it was released, now recognizes that it may provide them with unexpected benefit. They'll need a few enhancements to make it work in their world.
- All of this work is *software maintenance*

Software Maintenance

Maintainable Software

- 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 have uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not be around when changes must be made.
  - *Therefore, the design and implementation of the software must "assist" the person who is making the change*

People
Innovation
Excellence

Software Maintenance

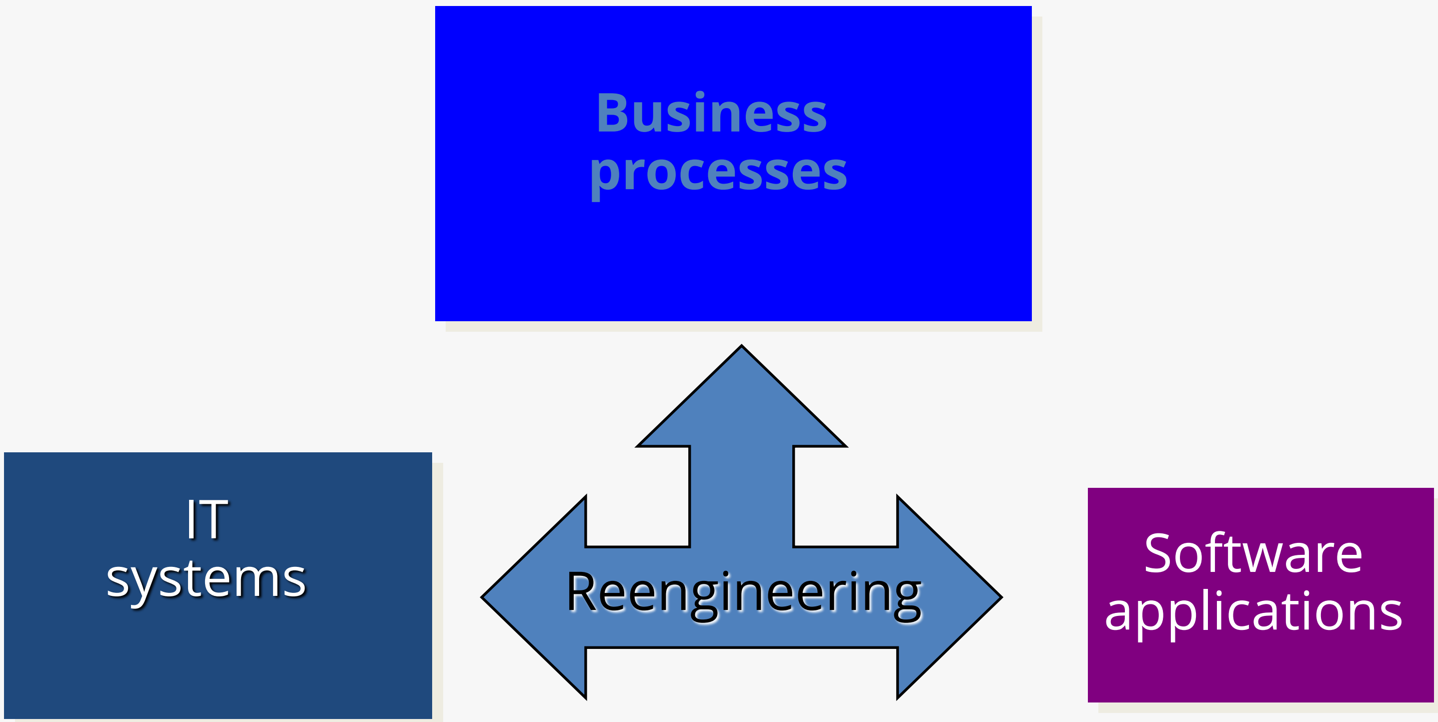
Software Supportability

- “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.” [SSO08]
- The software should contain facilities to assist support personnel when a defect is encountered in the operational environment (and make no mistake, defects *will* be encountered).
- Support personnel should have access to a database that contains records of all defects that have already been encountered—their characteristics, cause, and cure.

People
Innovation
Excellence

Business Process Reengineering

Reengineering

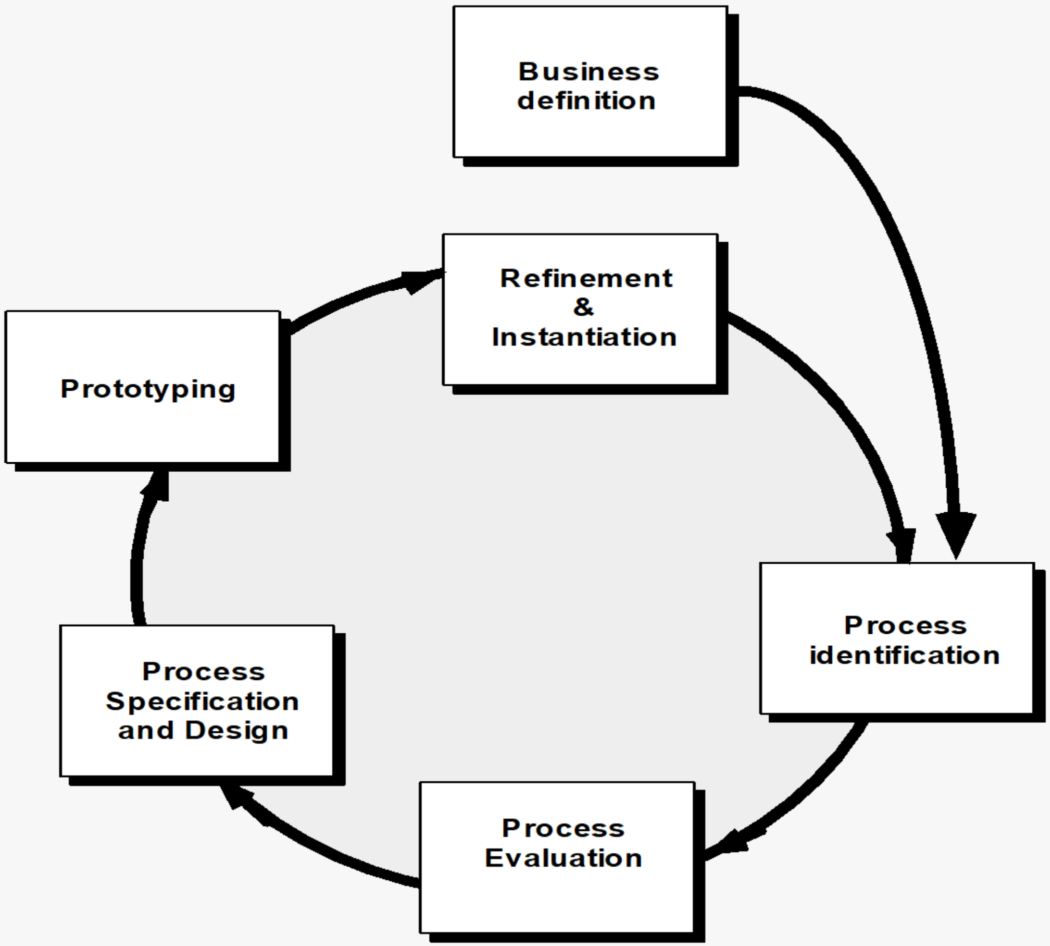


People
Innovation
Excellence

Business Process Reengineering

- **Business definition.** Business goals are identified within the context of four key drivers: cost reduction, time reduction, quality improvement, and personnel development and empowerment.
- **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified.
- **Process evaluation.** The existing process is thoroughly analyzed and measured.
- **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.
- **Prototyping.** A redesigned business process must be prototyped before it is fully integrated into the business.
- **Refinement and instantiation.** Based on feedback from the prototype, the business process is refined and then instantiated within a business system.

Business Process Reengineering



People
Innovation
Excellence

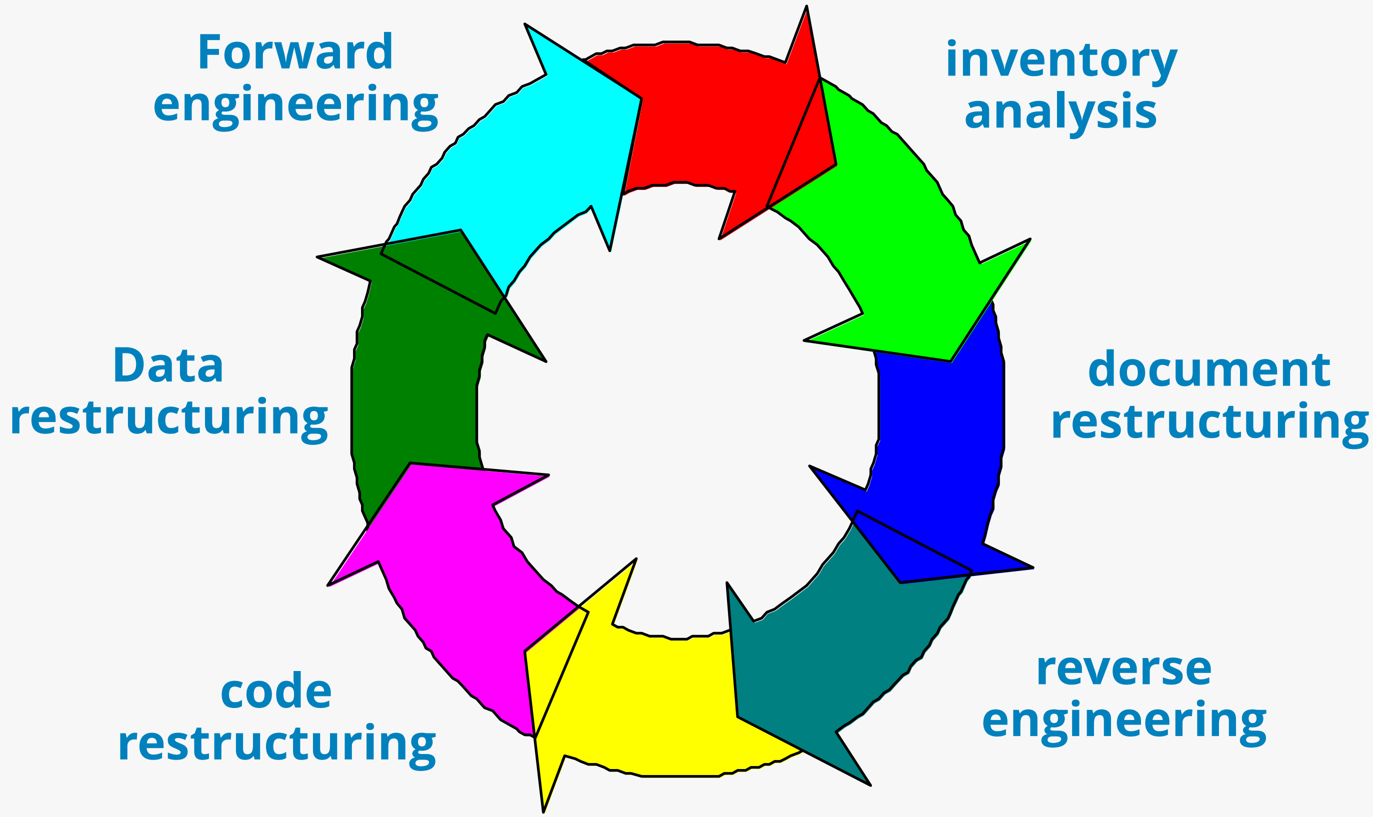
Business Process Reengineering

People
Innovation
Excellence

BPR Principles

- Organize around outcomes, not tasks.
- Have those who use the output of the process perform the process.
- Incorporate information processing work into the real work that produces the raw information.
- Treat geographically dispersed resources as though they were centralized.
- Link parallel activities instead of integrated their results.   When different
- Put the decision point where the work is performed, and build control into the process.
- Capture data once, at its source.

Software Reengineering



People
Innovation
Excellence

Software Reengineering

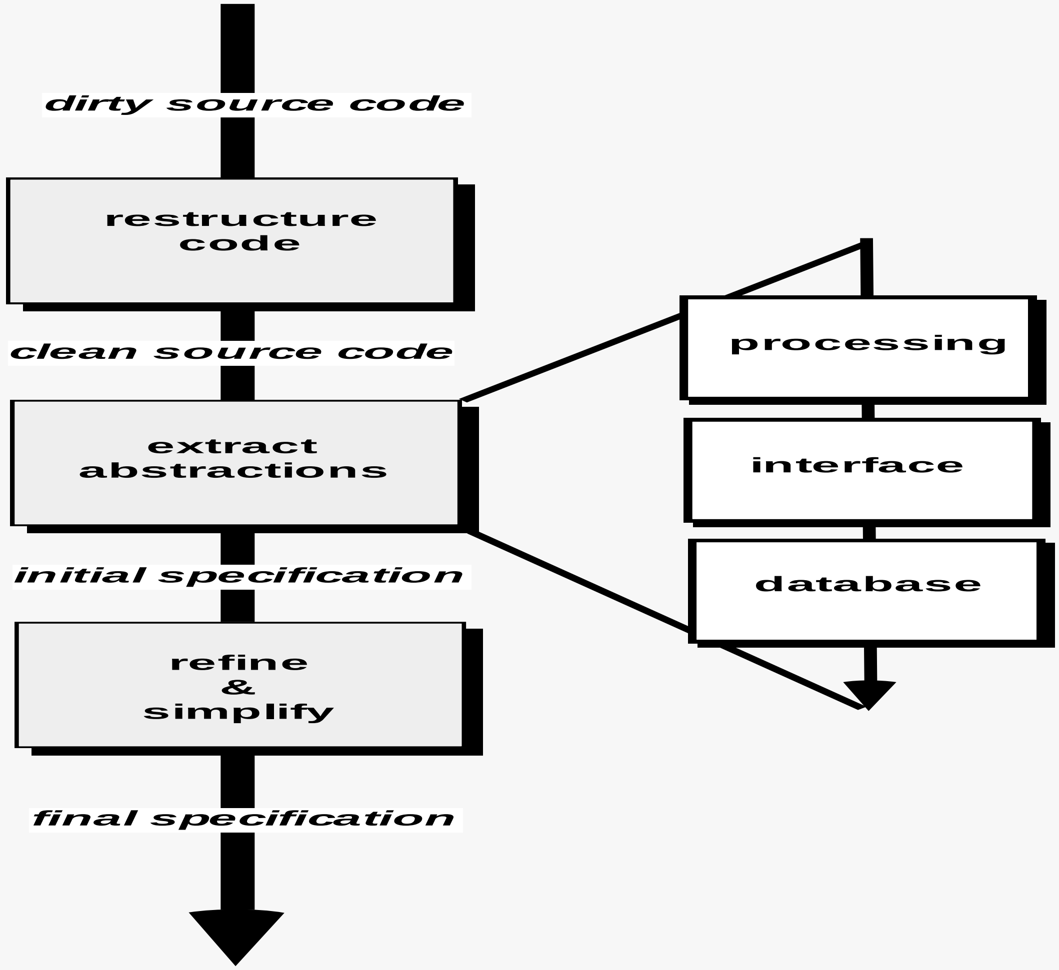
Inventory Analysis

- 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 change
  - effort applied to make the last change
  - system(s) in which it resides
  - applications to which it interfaces, ...
- analyze and prioritize to select candidates for reengineering

Document Restructuring

- Weak documentation is the trademark of many legacy systems.
- But what do we do about it? What are our options?
- Options …
  - *Creating documentation is far too time consuming.* If the system works, we'll live with what we have. In some cases, this is the correct approach.
  - *Documentation must be updated, but we have limited resources.* We'll use a "document when touched" approach. It may not be necessary to fully redocument an application.
  - *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.

Reverse Engineering



Reverse Engineering

Code Restructuring

- Source code is analyzed using a restructuring tool.
- Poorly design code segments are redesigned
- 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.

Reverse Engineering

Data Restructuring

- Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring 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 (e.g., flat files are currently implemented, when a relational approach would greatly simplify processing), 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.

People
Innovation
Excellence

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. CASE tools for reengineering will automate some parts of the job.
6. A complete software configuration (documents, programs and data) will exist upon completion of preventive maintenance.

BINUS UNIVERSITY

Forward Engineering

Economics of Reengineering-I

- 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 operation 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.

People
Innovation
Excellence

Forward Engineering

Economics of Reengineering-II

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

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

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

$$C_{reeng} = [P_6 - (P_4 + P_5) \times (L - P_8) - (P_7 \times P_9)]$$

- Using the costs presented in equations above, the overall benefit of reengineering can be computed as

$$\text{cost benefit} = C_{reeng} - C_{maint}$$

People
Innovation
Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Risk Management & operational Risk

  http://www.grafp.com/products/risk-manage.html
- Sw maintenance and Reengineering

  http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/13.25.SWEng4/html/text.html

People
Innovation
Excellence

Q & A

People
Innovation
Excellence