ICT-5405

PROJECT MANAGEMENT AND QUALITY ASSURANCE

09

Software Maintenance

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

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.

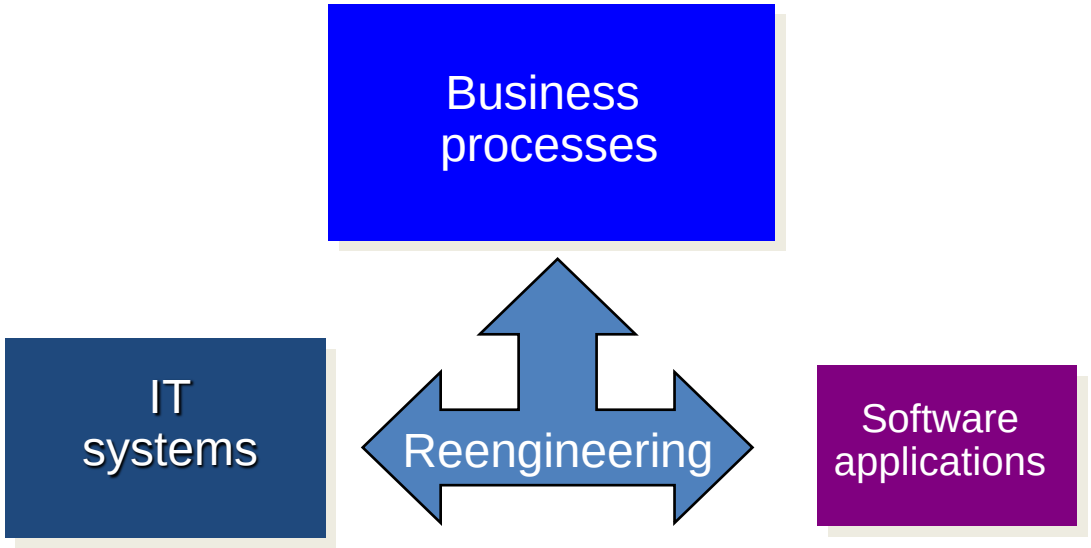
  Therefore, the design and implementation of the software must "assist" the person who is making the change

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.

Reengineering



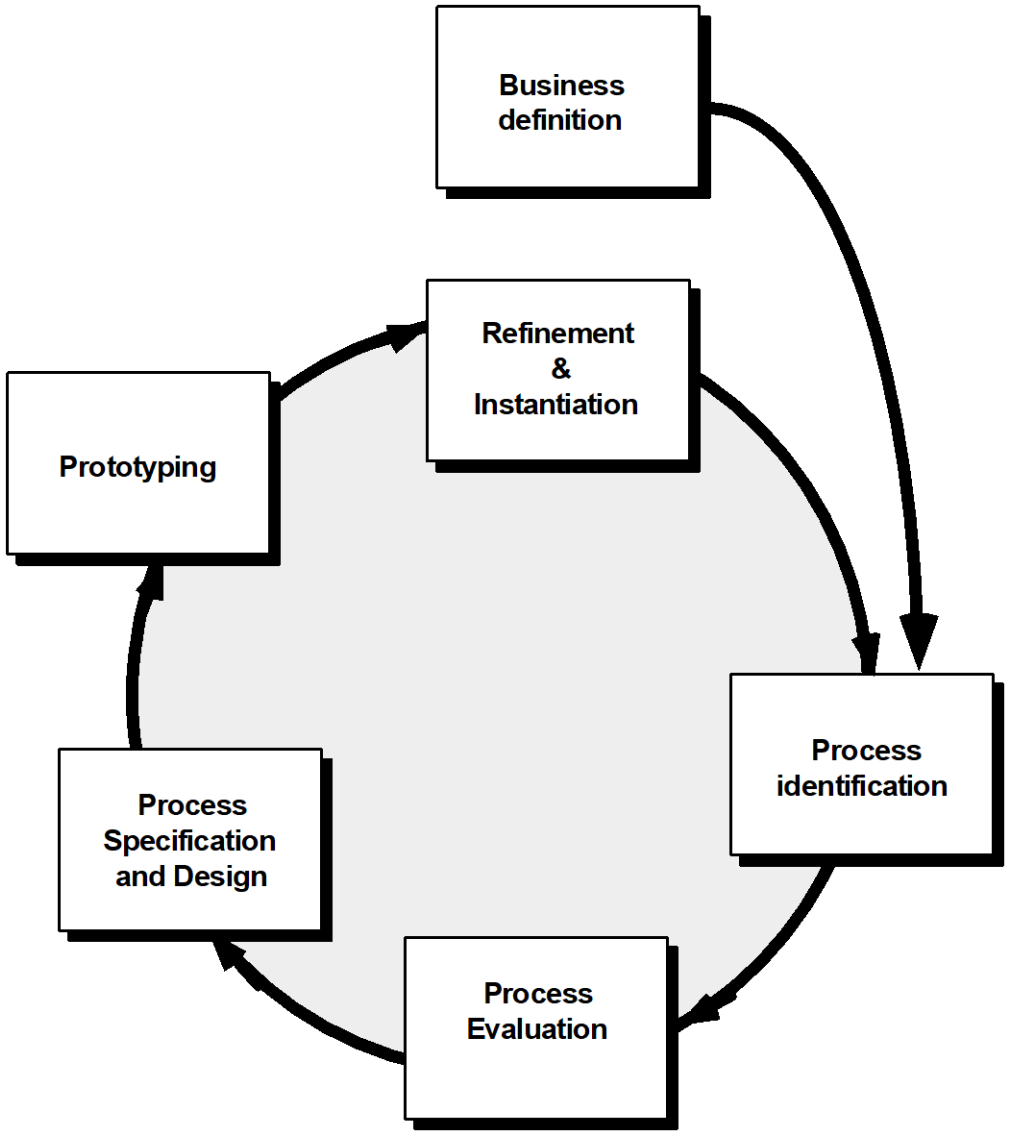
Business Processes

- A business process is “a set of logically related tasks performed to achieve a defined business outcome” [Dav90].
- 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

Business Process Reengineering(BPR) Model

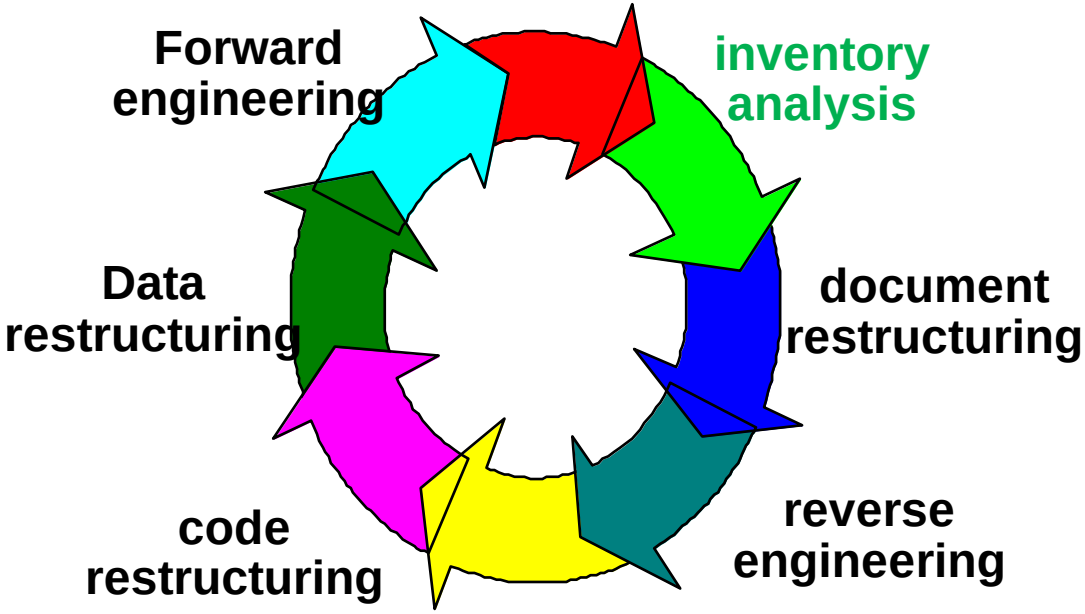
- **Business definition.** Business goals are identified within the context of four key drivers: cost reduction, time reduction, quality improvement, and personnel development and empowerment.
- **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified. They may then be ranked by importance, by need for change, or in any other way that is appropriate for the reengineering activity.
- **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.

BPR Model


Business definition
Refinement & Instantiation
Prototyping
Process identification
Process Specification and Design
Process Evaluation

Software Reengineering

- Reengineering is a rebuilding activity.



Inventory Analysis

- Every software organization should have an inventory of all applications.
- The inventory can be nothing more than a spreadsheet model containing information that provides a detailed description (e.g., size, age, business criticality) of every active application.
- By sorting this information according to business criticality, longevity, current maintainability and supportability, and other locally important criteria, candidates for reengineering appear.
- Resources can then be allocated to candidate applications for reengineering work.

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. Rather, those portions of the system that are currently undergoing change are fully documented. Over time, a collection of useful and relevant documentation will evolve.

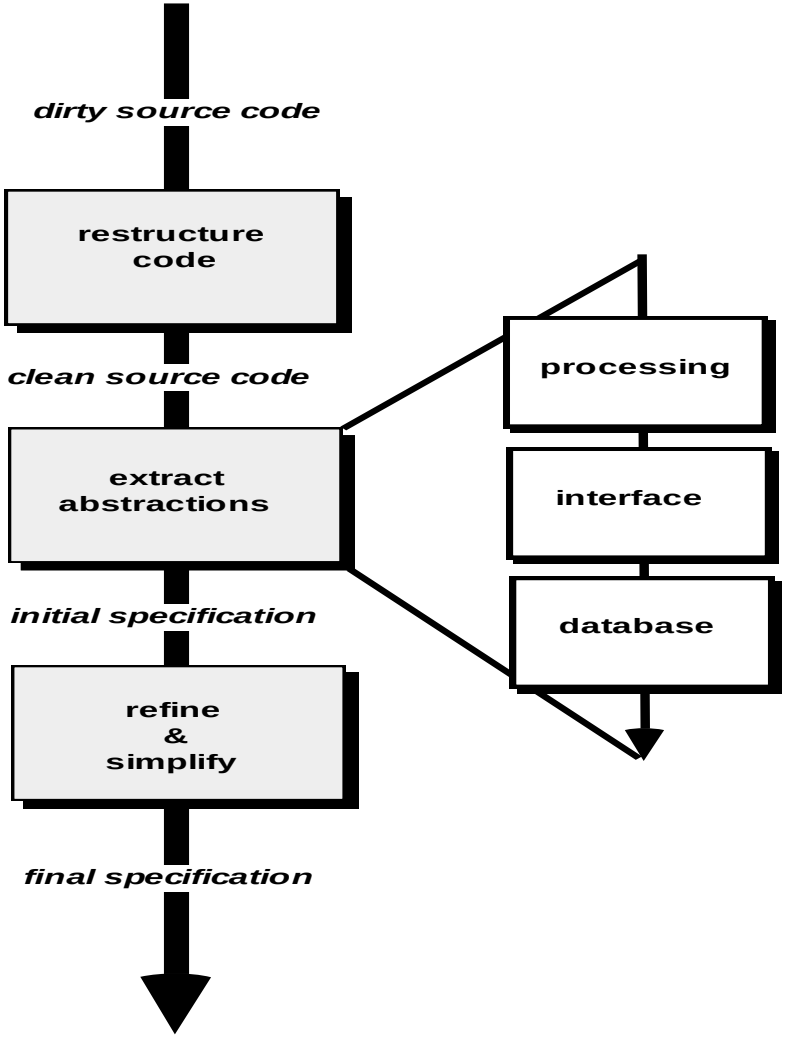
  The system is business critical and must be fully redocumented. Even in this case, an intelligent approach is to trim documentation to an essential minimum.

Reverse Engineering

- The term reverse engineering has its origins in the hardware world. A company disassembles a competitive hardware product in an effort to understand its competitor's design and manufacturing "secrets."
- These secrets could be easily understood if the competitor's design and manufacturing specifications were obtained. But these documents are proprietary and unavailable to the company doing the reverse engineering. In essence, successful reverse engineering derives one or more design and manufacturing specifications for a product by examining actual specimens of the product.

- Reverse engineering for software is quite similar. In most cases, however, the program to be reverse engineered is not a competitor's. Rather, it is the company's own work (often done many years earlier). The "secrets" to be understood are obscure because no specification was ever developed.
- Therefore, 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**. Reverse engineering tools extract data, architectural, and procedural design information from an existing program.

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.

Data Restructuring

- A program with weak data architecture will be difficult to adapt and enhance. In fact, 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 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.

Forward Engineering

- Forward engineering not only recovers design information from existing software but uses this information to alter or reconstitute the existing system in an effort to improve its overall quality.
- Redesign of the software architecture (program and/or data structure), using modern design concepts, can greatly facilitate future maintenance.
- Because a prototype of the software already exists, development productivity should be much higher than average.
- The user now has experience with the software. Therefore, new requirements and the direction of change can be ascertained with greater ease.
- CASE tools for reengineering will automate some parts of the job.
- In most cases, reengineered software reimplements the function of the existing system and also adds new functions and/or improves overall performance.