Implementation, Installation and After

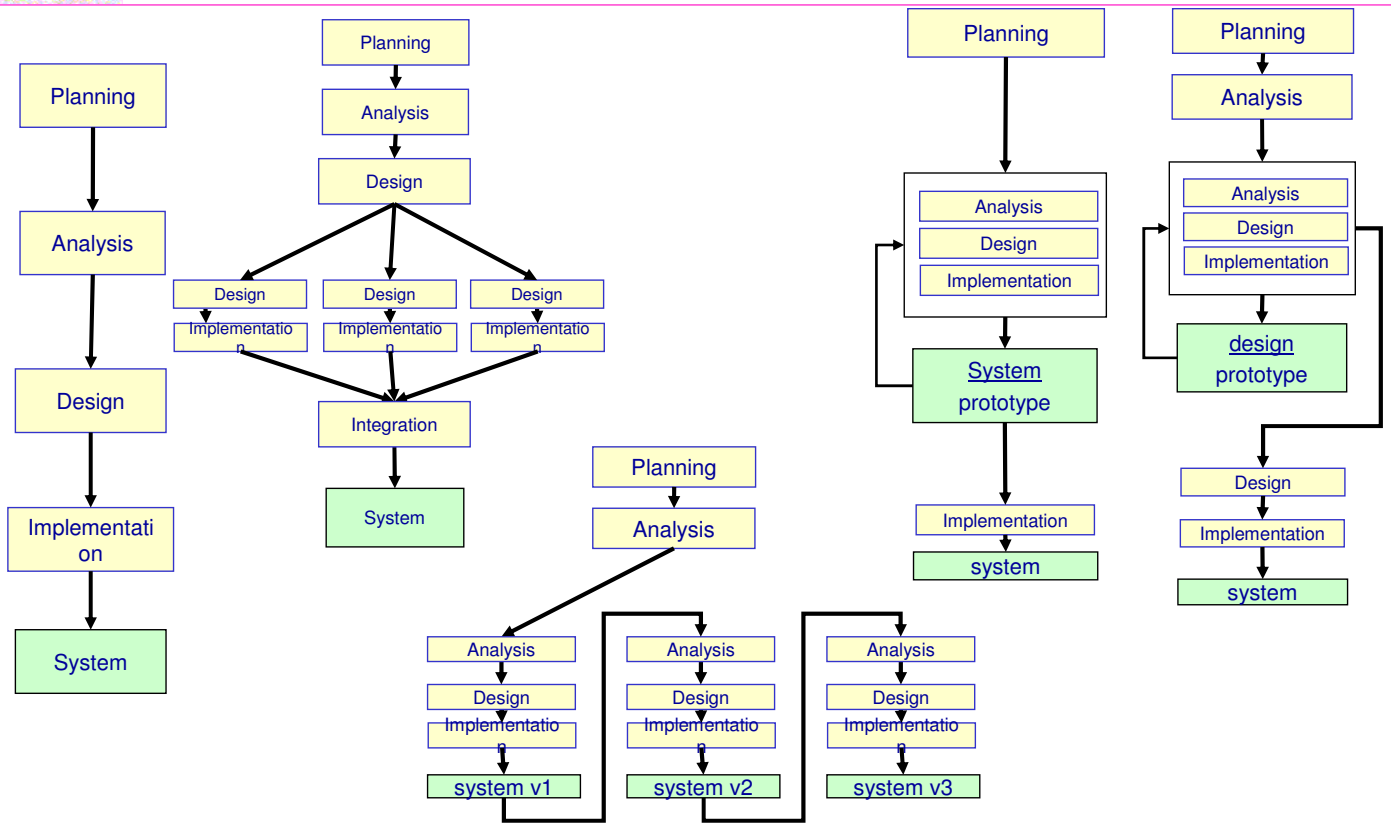
Lecture : 19
Date : 20-12-2005

Yannis Tzitzikas
University of Crete, Fall 2005

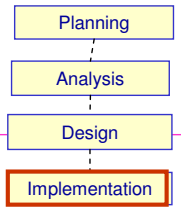
Outline

- The Implementation phase
- Managing Programming
- Testing
- Documentation
- Reuse
- Transition
- Transition management
- System support & maintenance
- Project assessment

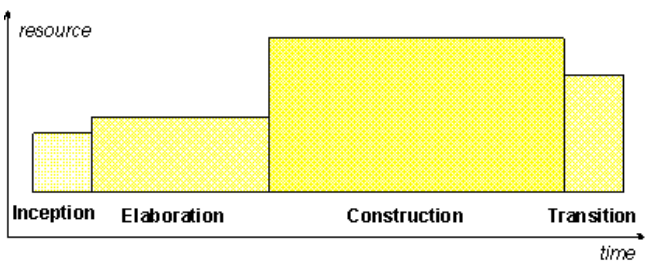
Refresher: Implementation phase and Development Methodologies



Implementation Phase



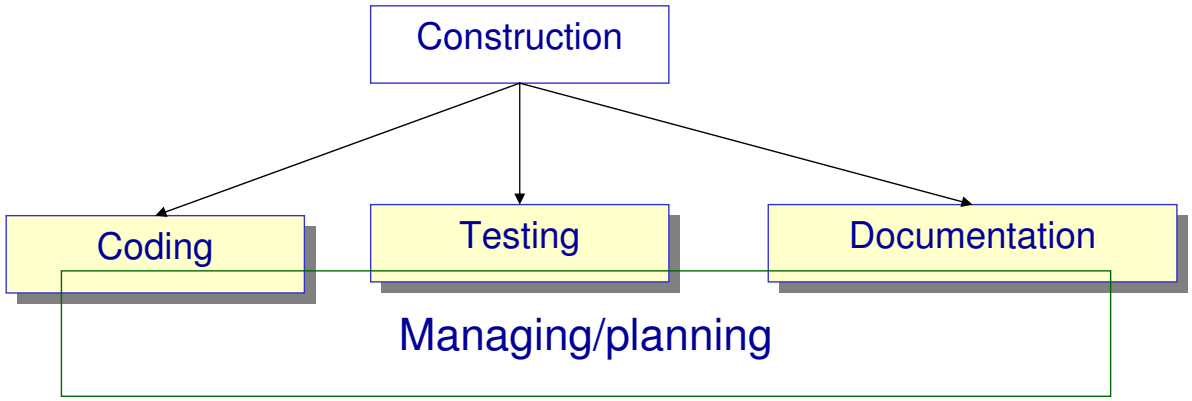
- Commonly, this is the longest and most expensive part of the process.
- Steps
  - 1. Construction
  - 2. Installation: The new system replaces the existing one
  - 3. Support Plan: formal and informal post-implementation review, as well as a systematic way for identifying changes needed for the system



Software Implementation

The implementation of a system requires a range of tools

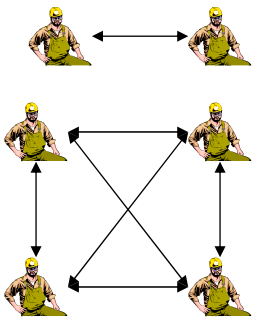
- CASE tools (code generators, reverse engineering tools)
- Compilers, interpreters and run-time support
- Visual Editors
- Integrated Development Environments (IDEs)
- Configuration management (e.g. CVS, RCS)
- Class browsers, Component Managers
- DBMSs (e.g. JDBC software to be installed at client-side)
- CORBA (IDL compiler)
- Testing Tools
- Installation Tools
- Conversion Tools (for transferring data from the old to the new system, e.g. Data Junction)
- Documentation generators



Managing Programming

Managing Programming

- The Programmer Paradox
  - After an appropriate number of people are assigned to a programming task, adding more people slows down rather than speeds up completion of the project.
- If a project is so complex that requires having a large team, the best strategy is to break the project into a series of smaller parts that can function as independently as possible.



Tasks

- Assigning the programmers
  - e.g. assign to each one a set of classes or a package
- Coordinating the activities
  - e.g. weekly meetings, coding standards, coding using three areas: development area, testing area, production area
- Managing the schedule

Managing the Schedule

- Use initial time estimates as a baseline
- Revise time estimates as construction proceeds
- Fight against scope creep
- Monitor “minor” slippage (one day here, one day there,... => behind schedule)
- Create risk assessment and track changing risks
- Fight the temptation to lower quality to meet unreasonable schedule demands

Avoid Classic Mistakes

1. Research-oriented development => if you use state-of-the art technology, lengthen planned time
2. Using “low-cost” personnel => lengthen planned time
3. Lack of code control (different persons work on the same code)
4. Inadequate testing => allocate sufficient time for testing

Testing

Testing

Testing is a form of insurance.

It costs more to repair software bugs when people are depending on the programs than in earlier stages before the systems are in use.

- *The purpose is not to demonstrate that the system is free of errors;*
- *The purpose is to detect as many errors as possible … :)*

Testing Philosophy

- It is dangerous to test early modules without an overall testing plan
- It may be difficult to reproduce sequence of events causing an error
- Testing must be done systematically and results documented carefully

Types/Stages of Testing

- **Unit testing**
  - Tests each module to assure that it performs its function
- **Integration testing**
  - Tests the interaction of modules to assure that they work together
- **System testing**
  - Tests to assure that the software works well as part of the overall system
- **Acceptance testing**
  - Tests to assure that the system serves organizational needs

Unit Testing

- Black Box Testing
  - Focuses on whether the unit meets requirements stated in specification
- White-Box Testing
  - Looks inside the module to test its major elements

Integration Testing

- User interface testing
  - Tests each interface function
- Use-case testing
  - Ensures that each use case works correctly
- Interaction testing
  - Tests each process in a step-by-step fashion
- System interface testing
  - Ensures data transfer between systems

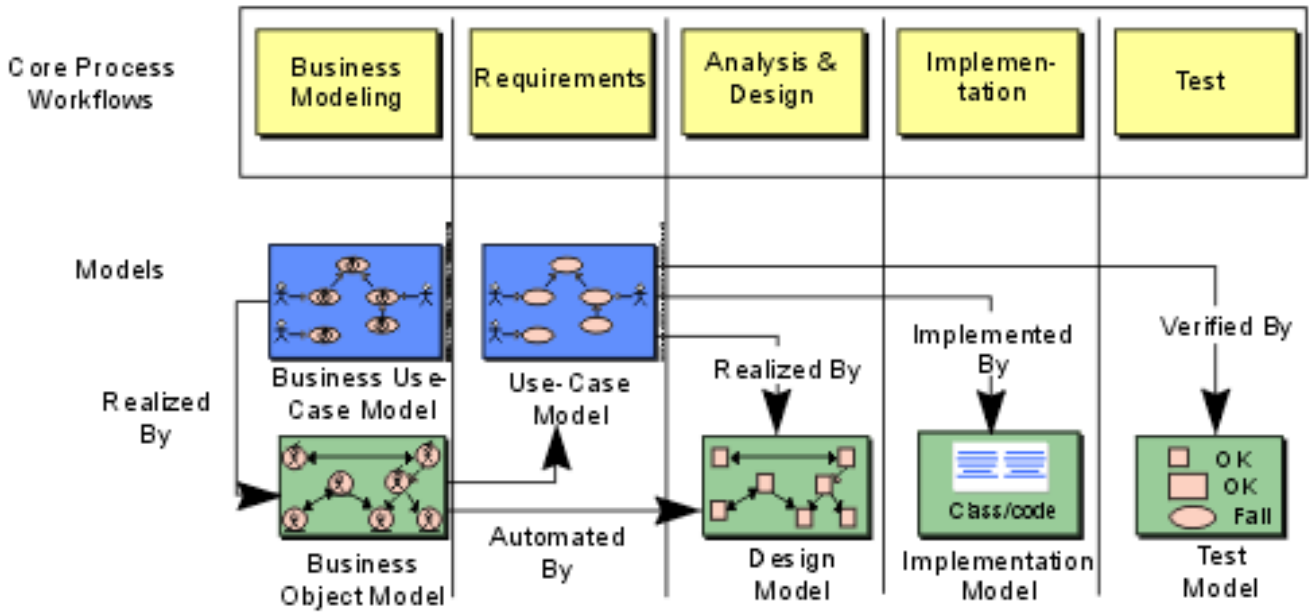
System Testing

- Requirements Testing
  - Ensures that integration did not cause new errors
- Usability Testing
  - Tests how easy and error-free the system is in use
- Security Testing
  - Assures that security functions are handled properly
- Performance Testing
  - Assures that the system works under high volumes of activity
- Documentation Testing
  - Analysts check that documentation and examples work properly

Acceptance Testing

- Alpha Testing
  - Repeats tests by users to assure they accept the system
- Beta Testing
  - Uses real data, not test data

Testing



Documentation

User Documentation

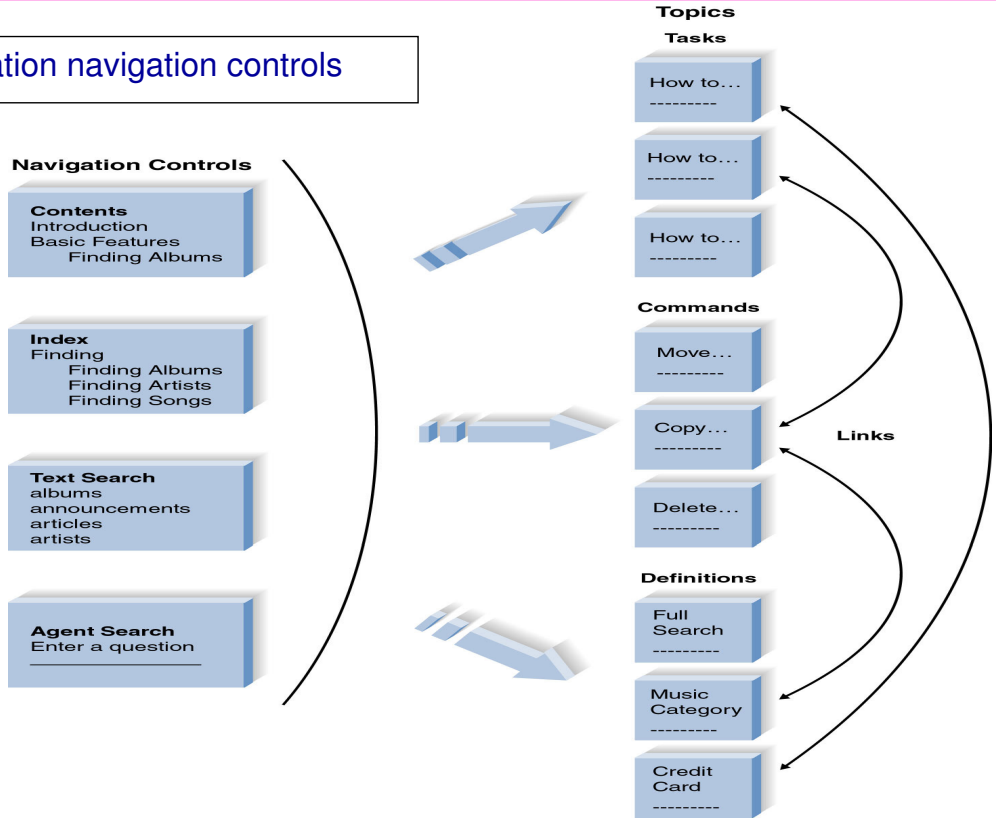
- Intended to help users operate the system
- High quality documentation takes about 3 hours per page
- The task should not be left to the end of the project
- Time required to develop and test user documentation should be built into project plan
- On-line documentation is growing in importance

Types of User Documentation

- **Reference documents**
  - i.e. “Help”. It helps users to use the system
- **Procedures manuals**
  - It help users to perform business tasks using the system (one business process may requires performing several system-supported tasks)
- **Tutorials**

Organizing On-Line Reference Documents

Documentation navigation controls

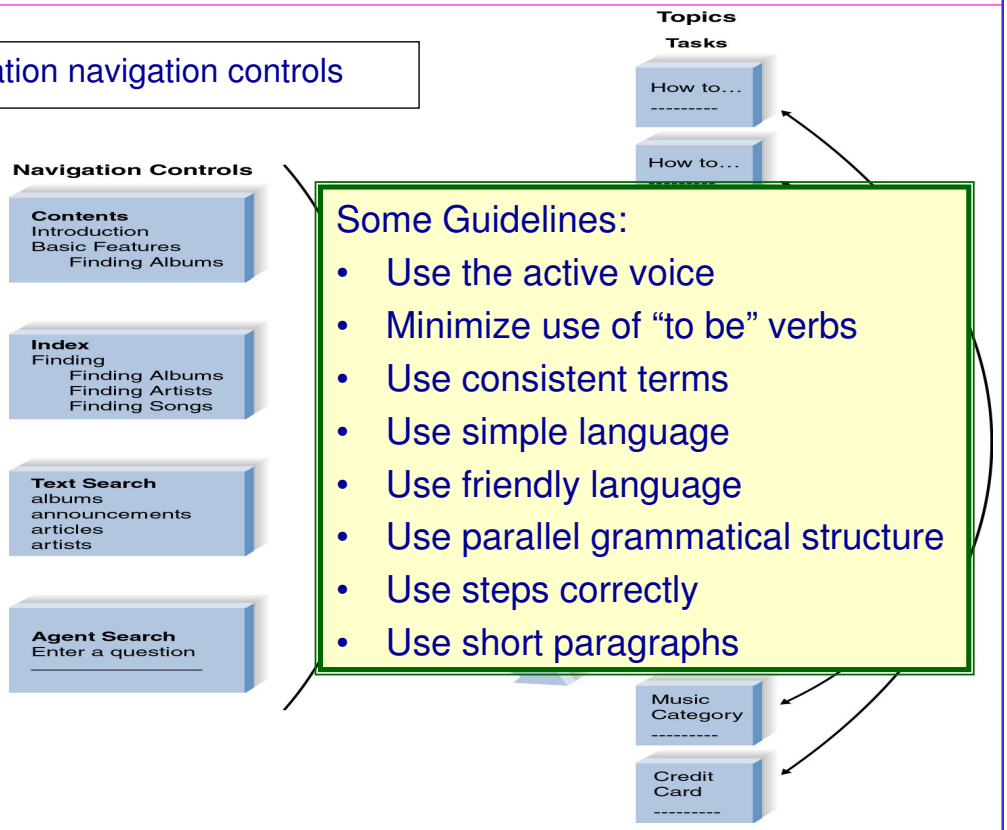


Documentation topics

- Commands and menus
- Common tasks
- Definitions

Organizing On-Line Reference Documents

Documentation navigation controls



Documentation topics

- Commands and menus
- Common tasks
- Definitions

Reuse !

Why Reuse?

- Economic reasons
  - If we can reuse design or components we can save time and money
- Quality reasons
  - If we can reuse a design or component that has been tested and proved to work properly, then the quality of our system enchances
    - less risk

Recall that the main objective of Information System Analysis and Design is to build systems that meet the client's requirements given specific time and budget constraints.

Why we don't reuse a lot?

- Planning for reuse too late
  - If reuse is appropriate (for the project at hand) it is something that needs to be planned for even before a project starts. This requires having the appropriate people and tools in place to make it possible
- The level of coupling between different classes
  - we usually adapt the code we write to the current project (so the resulting code is not directly reusable)

Planning a Strategy for Reuse

The SELECT Perspective [Allen and Frost 1998]

- Repository-based component management software.
  - Components are placed in a repository as a means of publishing them and making them available to other users. The repository is made up of catalogues and the catalogues contain details of components, their specifications and their interfaces.
  - Component management software tools provide the functionality for adding components to the repository and for browsing and searching for components. These may be integrated with CASE tools to allow the storage of analysis and design models as well as source code and executables.

Planning a Strategy for Reuse (II)

Reuse-driven Software Engineering Business

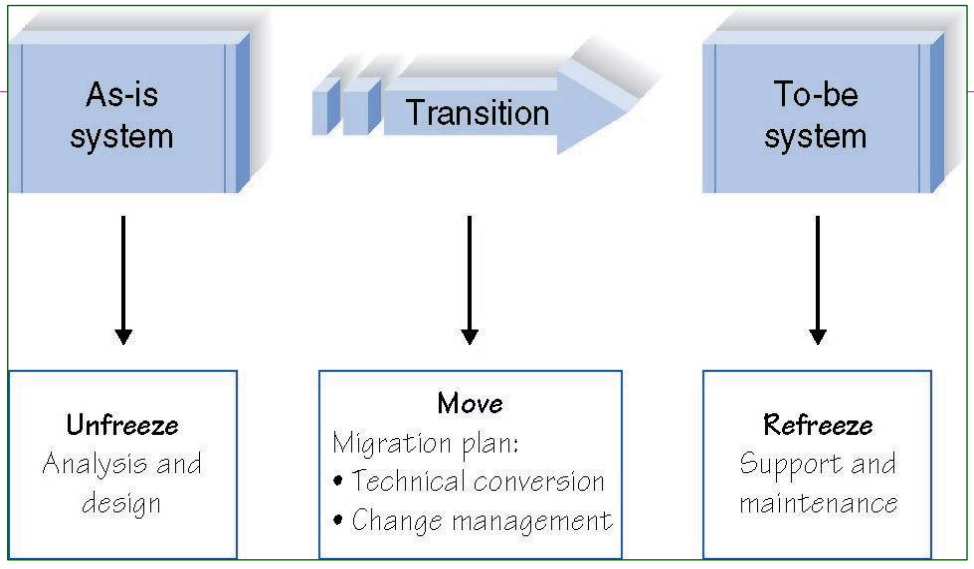
- key notions: reuse from the start
- Instead of considering components as executables (or as packages of executables designed to deliver a particular service), we can consider reuse in terms of any of the work products of systems development.
- So models used before coding are also subject to reuse:
  - Requirements Capture Unit
  - Design Unit
  - Testing Unit
  - Component Engineering Unit
  - Architecture Unit
  - Component Support Unit

Component Standards

Borland Delphi Microsoft Visual Basic	Object Pascal units compiled into .dll files - Dynamic Link Libraries .vbx files - Visual Basic Extensions .ocx files
Microsoft Windows	.ole files - Object Linking and Embedding DDE - Dynamic Data Exchange .dll files - Dynamic Link Libraries COM - Common Object Model DCOM - Distributed Common Object Model
CORBA	.idl files - Interface Definition Language IOP - Inter-ORB Protocol
Java	.jar files - Java Archive packages JavaBeans
Microsoft .NET	MSIL - Microsoft Intermediate Language CLR - Common Language Runtime WSDL - Web Service Description Language

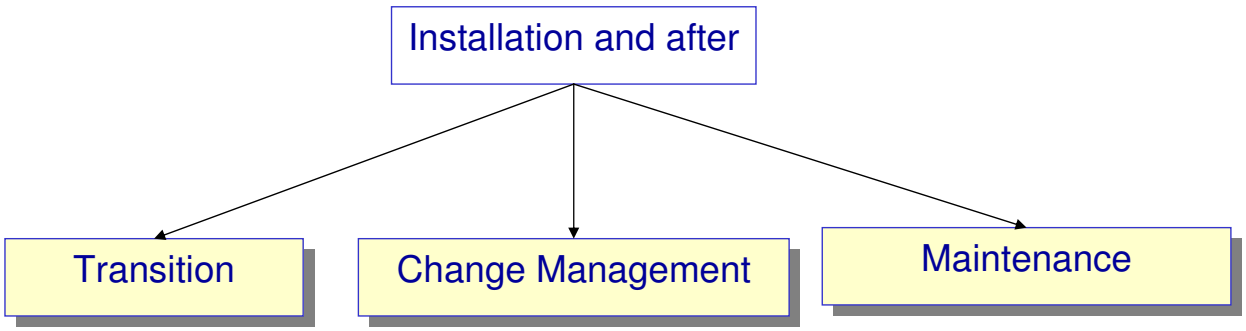
Installation and After

Transition



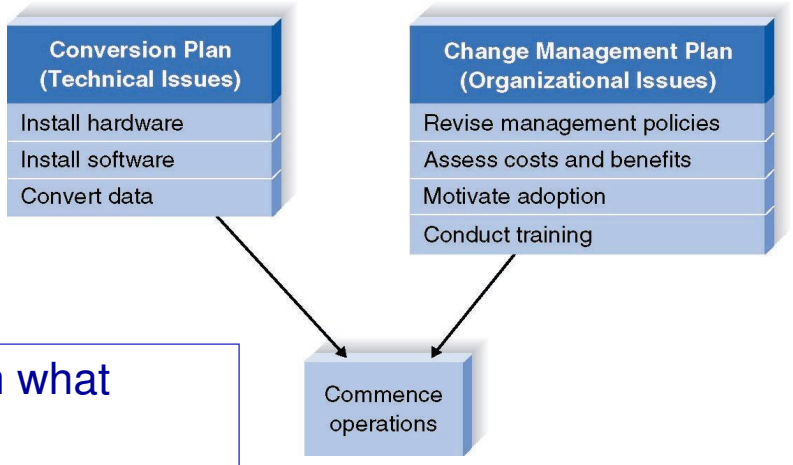
- Transitioning to new systems involves managing change from pre-existing norms and habits.
- Change management involves:
  - Unfreezing -- loosening up peoples' habits and norms
  - Moving -- transition from old to new systems
  - Refreezing -- institutionalize and make efficient the new way of doing things

Installation and After



Transition

Migration Plan



It describes who will perform what

- Technical aspects
  - Installing hardware and software
  - Converting data
- Organizational aspects
  - Training users on the system
  - Motivating employees to use the new system to aid in their work

Migration Plan

Style

- Direct: the new system instantly replaces the old
- Parallel: for a period of time both (old and new) systems are used. The old is turned off when the new is proven fully capable

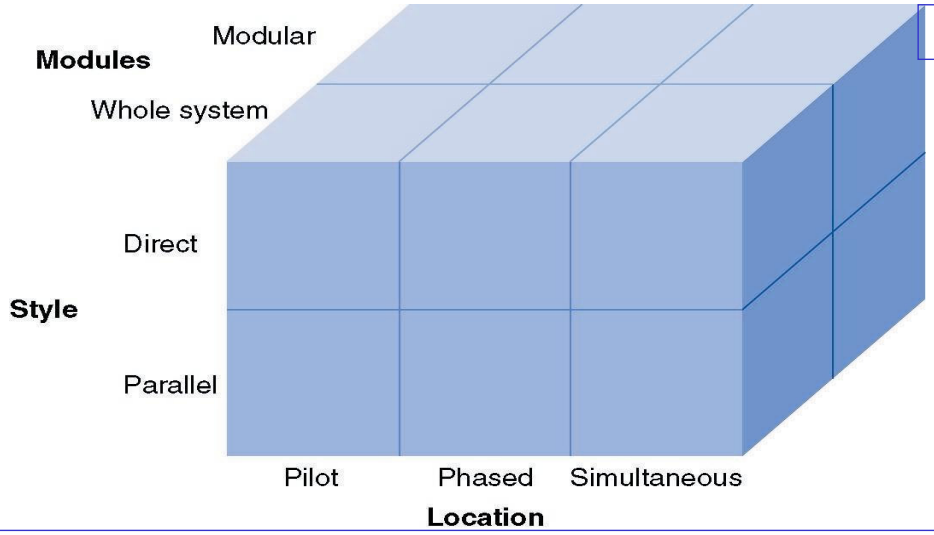
Location

- Pilot: One or more locations are converted to work out bugs
- Phased: Locations are converted in sets
- Simultaneous: All locations are converted at the same time

Modules

- Whole system: All modules converted in one step
- Modular: If modules are loosely associated, they can be converted one at a time

Transition Strategies



Dennis at al. 2005

Factors for selecting the a conversion strategy:

- Risk: Seriousness of consequences of remaining bugs
- Cost: Parallel requires paying for two systems for a period of time. Simultaneous requires more staff to support all locations
- Time: Parallel, phased, and modular require more time

Change Management

The process of helping persons of the organization to adapt to the new system

Change Management

Key Roles
- The sponsor is the business person who initiated the request for the new system
- The change agent is the person(s) who lead the change effort

Understanding Resistance to Change
- What is good for the organization, is not necessarily good for the individuals who work there
- Adapting to new work processes requires effort, for which there may be no additional compensation

- No computer system will be successfully adopted unless management policies support its adoption
- Management tools for supporting adoption
  - Standard operating procedures (SOPs)
  - Measurements and rewards

Change Management (II)

Motivating Adoption
- The information strategy aims to convince adopters that change is better
- The political strategy uses organizational power to motivate change
- Differentiate between ready adopters, reluctant adopters, and resistant adopters

Training

- Every new system requires new skills
- New skills may involve use of the technology itself
- New skills may be needed to handle the changed business processes

What to Train

- Should focus on helping users accomplish their tasks
- Use cases provide an outline for common activities and a basis to plan training

Types of Training

- One-to-One
- Classroom
- Computer-based

Institutionalization of the System

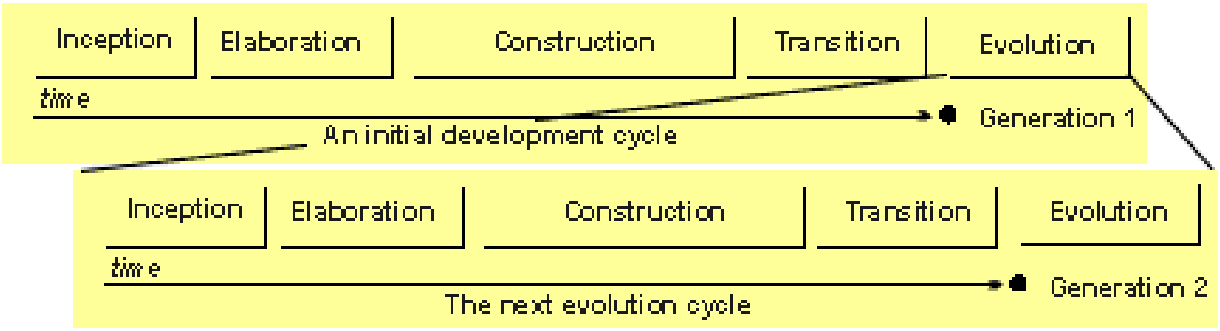
Types of System Support

- On-demand training at time of user need
- Online support
  - Frequently asked questions (FAQ)
- Help desk
  - Phone service for known issues

Sources of Change Requests

- Problem reports from the operations group
- Requests for enhancements from users
- Requests from other systems development projects
- Change requests from senior management

Post-implementation



Post Implementation Activities

- Provide support
  - Assistance in using the system
- Provide maintenance
  - Repair or fix discovered bugs or errors
  - Add minor enhancements to provide added value
- Assess the project
  - Analyze what was done well
  - Discover what activities need improvement in the future

Project Assessment

- Important for continued project improvement
- Especially important for junior personnel to improve quickly

Project Team Review

- Each member prepares 2-3 page document regarding her or his actions during the project
- Focus on improvement not penalties
- Excellent behaviors are acknowledged and diffused to others
- Team leader summarizes and distributes lessons learned

System Review

- Examine the extent to which the costs and benefits of the system are realized
- Use this information to help in more accurately estimating costs and benefits for future projects

Recall CMM (Lecture 2)