HY351:
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design

Implementation, Installation and After

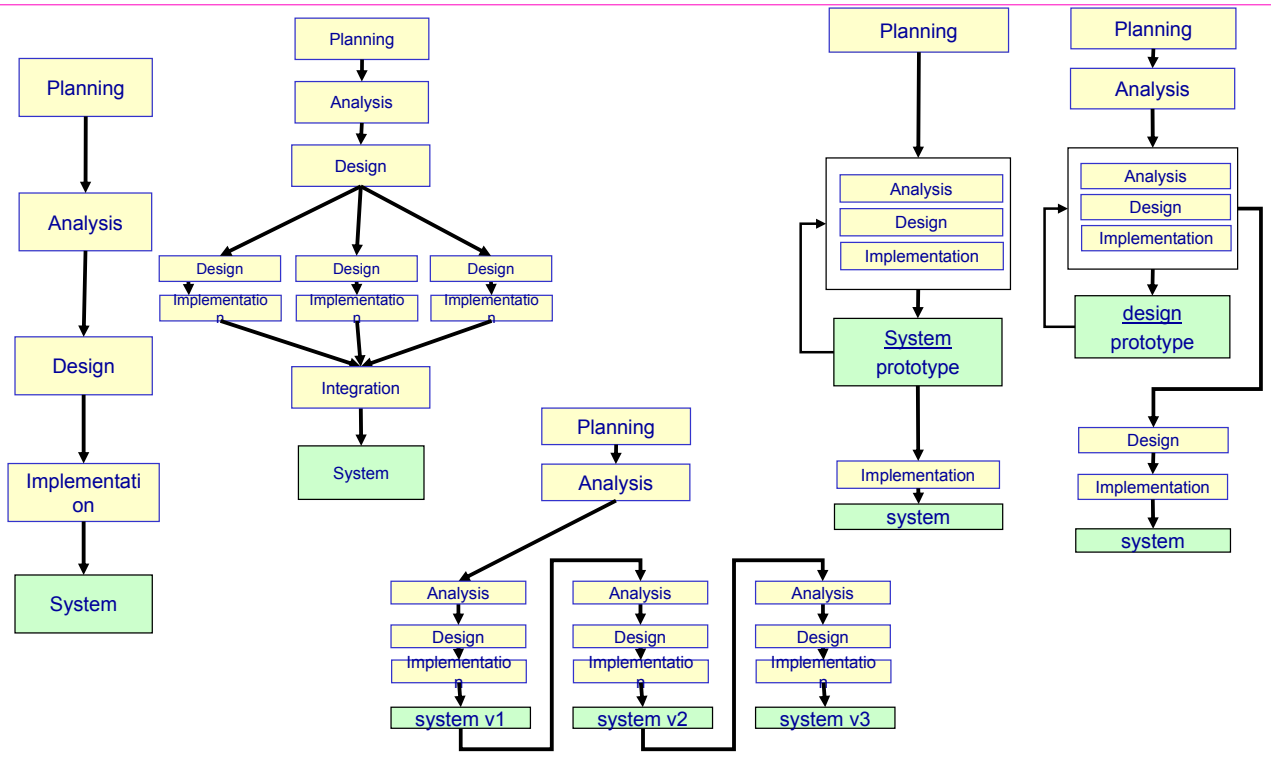
Γιάννης Τζίτζικας

Διάλεξη : 18
Ημερομηνία :
Θέμα :

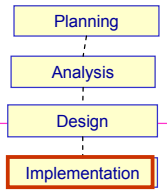
Outline

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

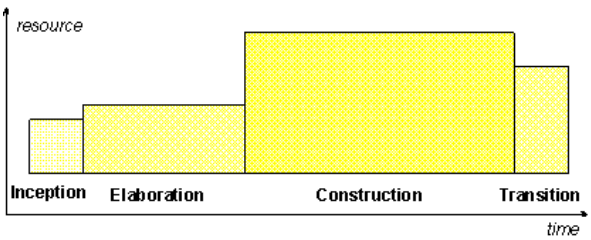
Refresher: Implementation phase and Development Methodologies



Implementation Phase



- Commonly, this is the **longest** and **most expensive** part of the process.
- General steps
  - 1. Construction
  - 2. Installation & Transition: 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. `SVN, CVS, RCS`)
- Class browsers, Component Managers
- DBMSs (e.g. `JDBC` software to be installed at client-side)
- CORBA (`IDL` compiler)
- Testing Tools (e.g. `JUnit`)
- Installation Tools
- Conversion Tools (for transferring data from the old to the new system, e.g. `Data Junction`)
- Documentation generators

Some indicative development tools

Development and Configuration Control Tools and Frameworks

- Ant (http://ant.apache.org/)
- Maven (maven.apache.org)
- SVN (http://subversion.tigris.org/)
- GForge (http://gforge.org/)
- Trac (http://trac.edgewall.org/)
- CruiseControl (http://cruisecontrol.sourceforge.net/)
- Hudson (https://hudson.dev.java.net/)

Ant

- Apache Ant is a software tool for automating software build processes. It is similar to make but is written in the Java language, requires the Java platform, and is best suited to building Java projects.
- The most immediately noticeable difference between Ant and make is that Ant uses XML to describe the build process and its dependencies, whereas make has its Makefile format. By default the XML file is named build.xml.

Ant

Sample build.xml file

This is about a simple Java "Hello, world" application. It defines three targets - *clean*, *compile* and *jar*, each of which has an associated description. The *jar* target lists the *compile* target as a dependency. This tells Ant that before it can start the *jar* target it must first complete the *compile* target.

```
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="compile"
     description="compile the Java source code to class files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile"
     description="create a Jar file for the application">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

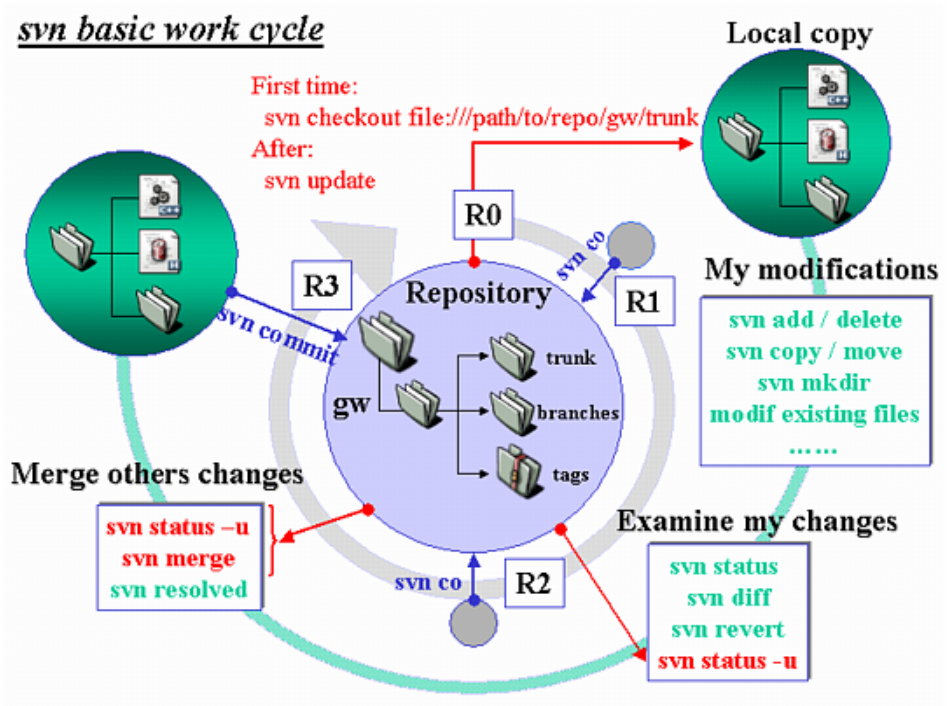

Maven (maven.apache.org)

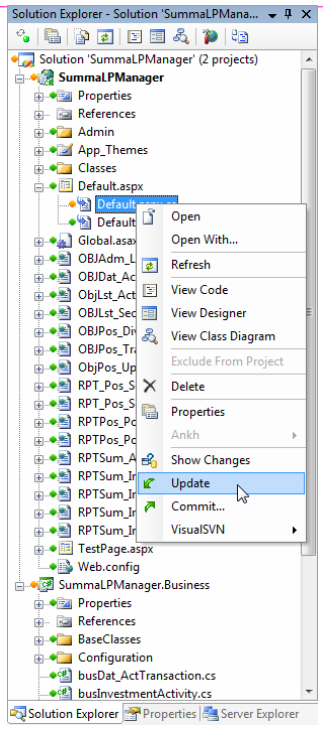
- Maven is a software tool for Java programming language project management and automated software build. It is similar in functionality to the Apache Ant tool (and to a lesser extent, PHP's PEAR and Perl's CPAN), but has a simpler build configuration model, based on an XML format. Maven is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.
- Maven uses a construct known as a Project Object Model (POM) to describe the software project being built, its dependencies on other external modules and components, and the build order. It comes with pre-defined targets for performing certain well defined tasks such as compilation of code and its packaging.
- A key feature of Maven is that it is network-ready. The core engine can dynamically download plug-ins from a repository, the same repository that provides access to many versions of different Open Source Java projects, from Apache and other organisations and developers. This repository and its reorganized successor, the Maven 2 repository, strives to be the de facto distribution mechanism for Java applications, but its adoption has been slow. Maven provides built in support not just for retrieving files from this repository, but to upload artefacts at the end of the build. A local cache of downloaded artefacts acts as the primary means of synchronizing the output of projects on a local system.
- See Maven in 5 minutes: http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

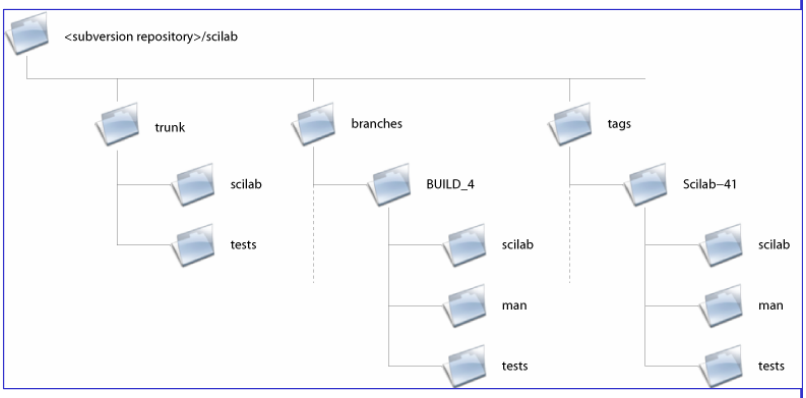
Subversion (svn)

- It could be used to share files bewteen several developers and to keep track of the history of a modified file.
- Subversion (svn) is a free/open source version control system. The central point is called a repository. The repository is like a file server that remembers every changes made to your files and directories. This allows you to recover old versions of your files or examine the history of how your files have changed. Svn can access a repository (equivalent to a directory) across networks which allows yours files to be shared/used/modified by people working on different computers. At the beginning, the revision number is 0 (R0). Any successful submission to the repository will increment this number by 1.
  The typical work cycle with subversion is illustrated on the next slides

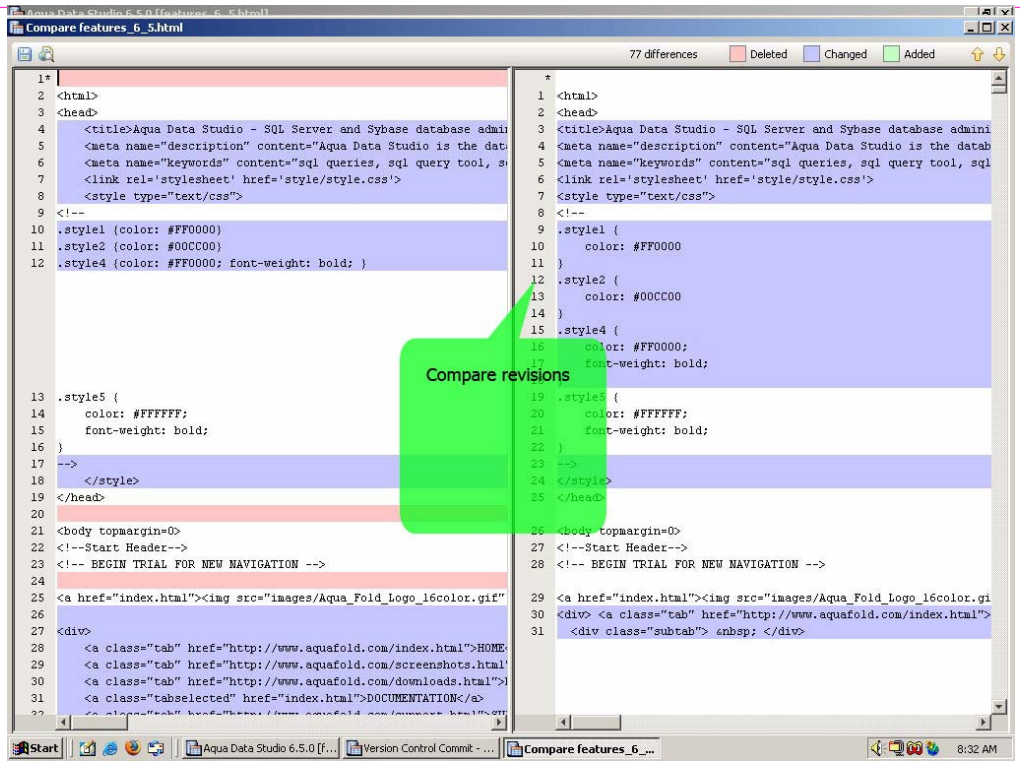
Subversion (svn)



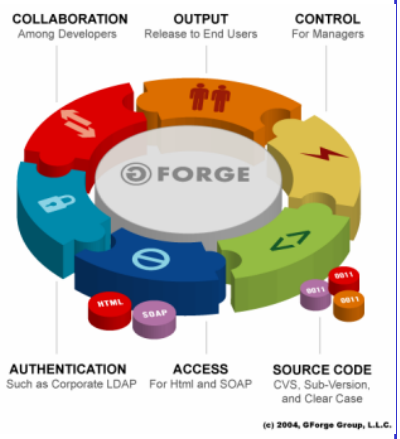

Solution Explorer - Solution 'SummaLPMana...
Solution 'SummaLPManager' (2 projects)
SummaLPManager
Properties
References
Admin
App_Themes
Classes
Default.aspx
Global.asax
TestPage.aspx
Web.config
SummaLPManager.Business
Properties
References
BaseClasses
Configuration
busDat_ActTransaction.cs
busInvestmentActivity.cs
Open
Open With...
Refresh
View Code
View Designer
View Class Diagram
Exclude From Project
Delete
Properties
Ankh
Show Changes
Update
Commit...
VisualSVN
Solution Explorer
Properties
Server Explorer


<subversion repository>/scilab
trunk
branches
tags
scilab
tests
BUILD_4
Scilab-41
scilab
man
tests
scilab
man
tests

comparing


Compare features_6_5.html
77 differences
Deleted
Changed
Added
Compare revisions
Start
Aqua Data Studio 6.5.0 [f...
Version Control Commit - ...
Compare features_6_...
8:32 AM

GFORGE

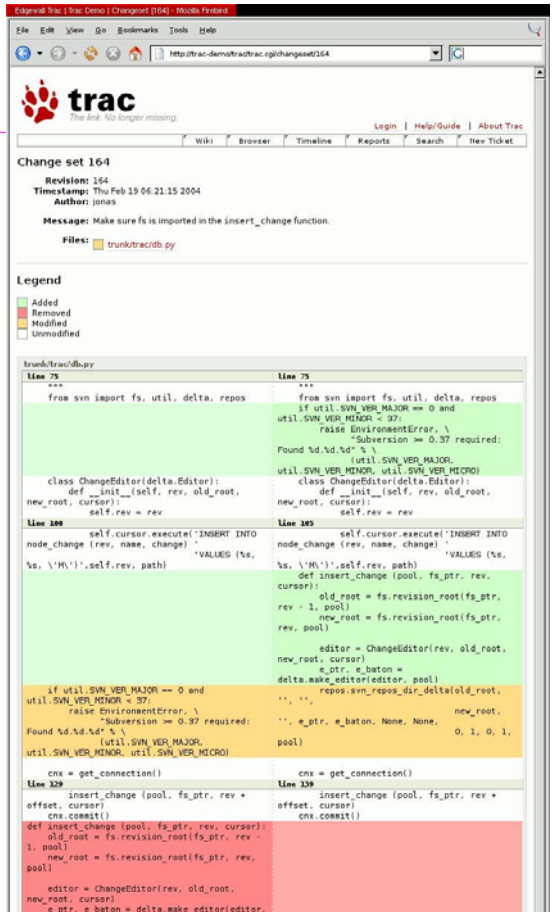


GForge can provide a centralized access point for several useful utilities and tools which could be used in a project. Some of these tools include:

- A version control repository (SVN)
- Mailing lists
- Discussion forums
- Bug tracking
- A web interface to SVN
- Task lists
- A website which provides some usage statistics, including the project members, the number of mailing lists, SVN statistics, the number of items in the discussion forums, etc

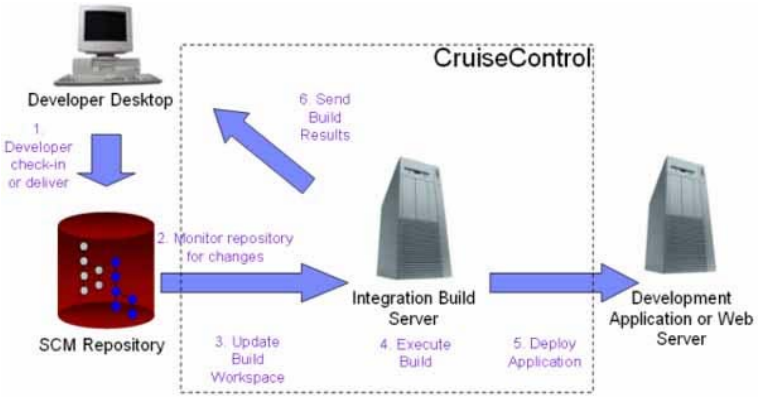
Trac

- Trac is an enhanced wiki and issue tracking system for software development projects. It provides an interface to Subversion, an integrated Wiki and convenient reporting facilities. It allows wiki markup in issue descriptions and commit messages, creating links and seamless references between bugs, tasks, changesets, files and wiki pages. A timeline shows all project events in order, making the acquisition of an overview of the project and tracking progress very easy.



cruisecontrol

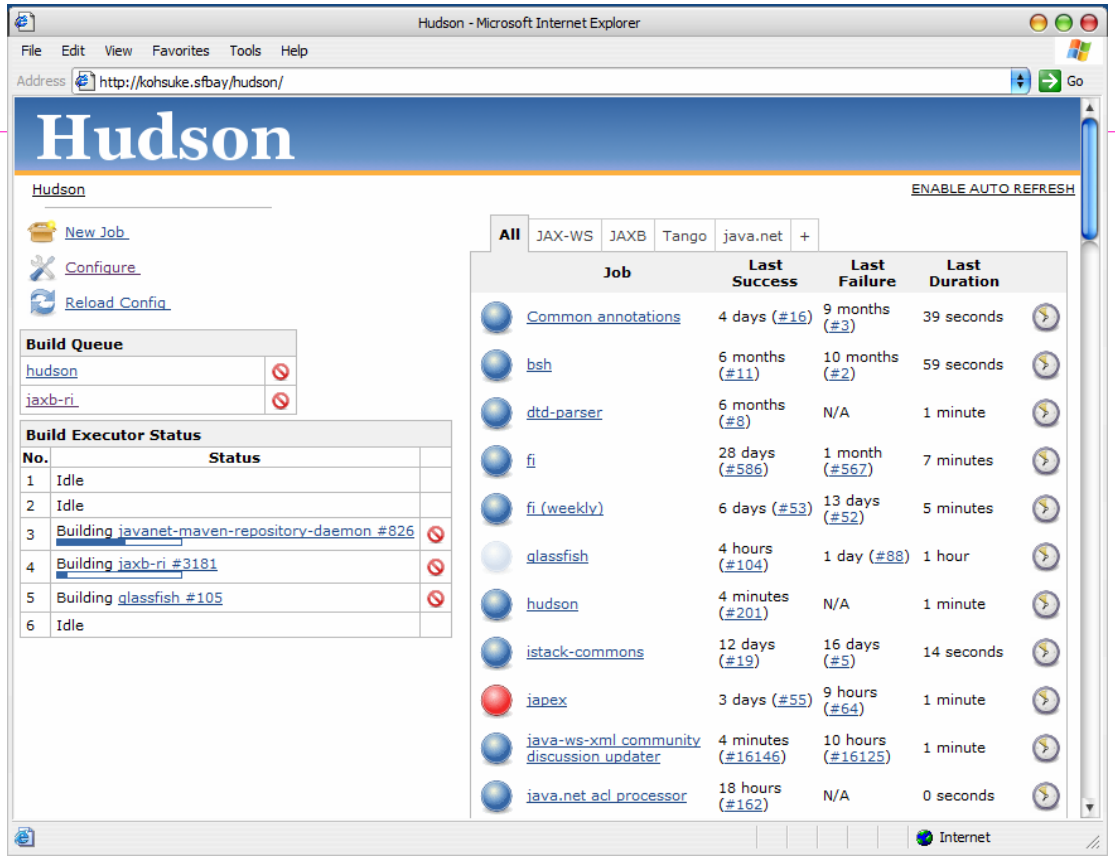
- CruiseControl is a framework for a continuous build process. It includes, but is not limited to, plugins for email notification, Ant, and various source control tools. A web interface is provided to view the details of the current and previous builds.
- CruiseControl is distributed under a BSD-style license and is free for use. CruiseControl adheres to an open source model and therefore makes the source code freely available.

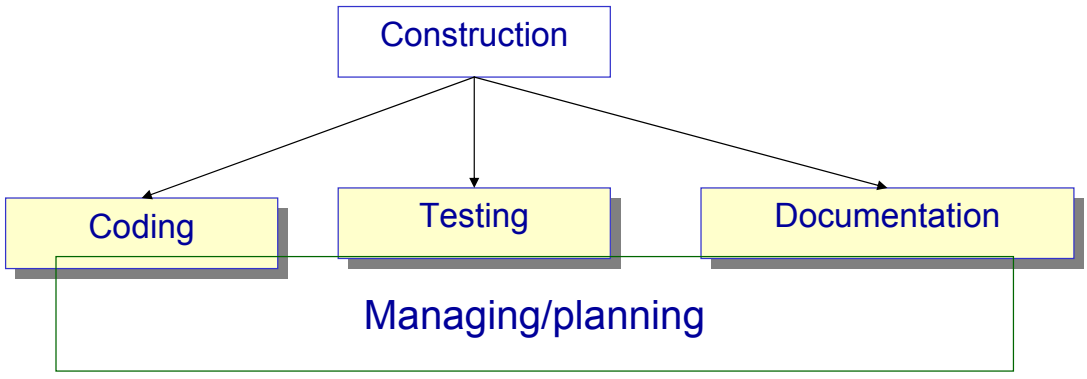


Hudson

Hudson monitors executions of repeated jobs, such as building a software project or jobs run by cron. Among those things, current Hudson focuses on the following two jobs:

- **Building/testing software projects continuously**, just like CruiseControl or DamageControl. In a nutshell, Hudson provides an easy-to-use so-called continuous integration system, making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. The automated, continuous build increases the productivity.
- **Monitoring executions of externally-run jobs**, such as cron jobs and procmail jobs, even those that are run on a remote machine. For example, with cron, all you receive is regular e-mails that capture the output, and it is up to you to look at them diligently and notice when it broke. Hudson keeps those outputs and makes it easy for you to notice when something is wrong.

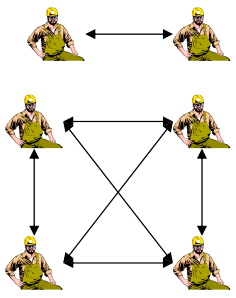

Hudson - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://kohsuke.sfbay/hudson/
Go
Hudson
Hudson
ENABLE AUTO REFRESH
New Job
Configure
Reload Config
Build Queue
hudson
jaxb-ri
Build Executor Status
No. Status
1 Idle
2 Idle
3 Building javanet-maven-repository-daemon #826
4 Building jaxb-ri #3181
5 Building glassfish #105
6 Idle
All JAX-WS JAXB Tango java.net +
Job Last Success Last Failure Last Duration
Common annotations 4 days (#16) 9 months (#3) 39 seconds
bsh 6 months (#11) 10 months (#2) 59 seconds
dtd-parser 6 months (#8) N/A 1 minute
fi 28 days (#586) 1 month (#567) 7 minutes
fi (weekly) 6 days (#53) 13 days (#52) 5 minutes
glassfish 4 hours (#104) 1 day (#88) 1 hour
hudson 4 minutes (#201) N/A 1 minute
istack-commons 12 days (#19) 16 days (#5) 14 seconds
japex 3 days (#55) 9 hours (#64) 1 minute
java-ws-xml community discussion updater 4 minutes (#16146) 10 hours (#16125) 1 minute
java.net acl processor 18 hours (#162) N/A 0 seconds
Internet


Construction
Coding
Testing
Documentation
Managing/planning

Managing Programming
(Διαχείριση Προγραμματισμού)

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

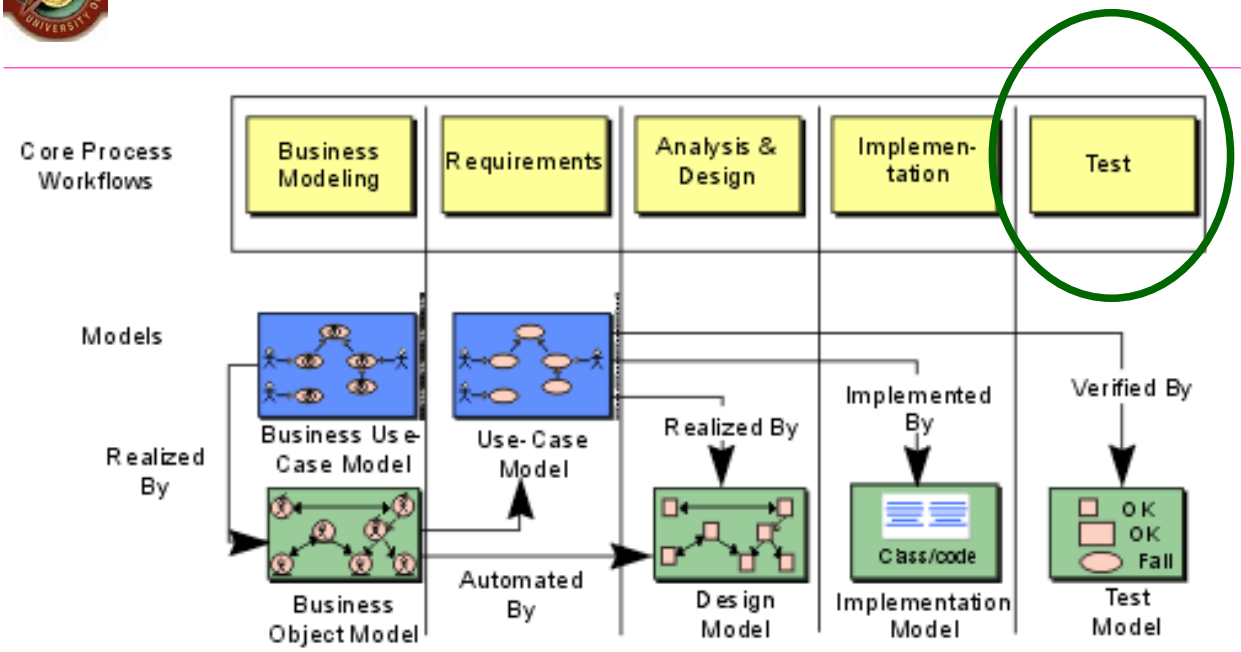
Managing the Schedule (Διαχείριση Χρονοδιαγράμματος)

- 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 (Έλεγχοι)



However it is a bad practice to leave testing for the end

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

Case study: Java Unit Testing

Case: Testing Java Code

- JUnit is a freely available Java unit test framework. extensively used in the Java community because of its elegance of design and ease of use.
- Why use a testing framework?
  - Using a testing framework is beneficial because it forces you to explicitly declare the expected results of specific program execution routes.
  - When debugging it is possible to write a test which expresses the result you are trying to achieve and then debug until the test comes out positive.
  - By having a set of tests that test all the core components of the project it is possible to modify specific areas of the project and immediately see the effect the modifications have on the other areas by the results of the test, hence, side-effects can be quickly realized.

- JUnit promotes the idea of first testing then coding, in that it is possible to setup test data for a unit which defines what the expected output is and then code until the tests pass. It is believed by some that this practice of "test a little, code a little, test a little, code a little..." increases programmer productivity and stability of program code whilst reducing programmer stress and the time spent debugging. It also integrates with Ant (http://jakarta.apache.org/ant)

Short Introduction to JUnit

Suppose you want to test the following code

```
public class Math {
     static public int add(int a, int b) {
         return a + b;
     }
}
```

```
import junit.framework.*;
public class TestMath extends TestCase {
  public void testAdd() {
          int num1 = 3;
          int num2 = 2;
          int total = 5;
          int sum = 0;
          sum = Math.add(num1, num2);
          assertEquals(sum, total);
     }
   }
```

To test this code, you need a second Java class that will

1) import junit.framework.*

2) extend TestCase.

Short Introduction to JUnit

- Notice that the routine is named testAddNumbers. This convention tells you that the routine is supposed to be a test and that it's targetting the "add" functionality.
- The last step is how to run your JUnit tests. You can do this several ways, including your command line, Eclipse or the JUnit Test Runner, or plain Ant. To run a Junit test with an Ant script, add this to your Ant script:

```
<junit printsummary="yes" haltonfailure="yes" showoutput="yes" >
          <classpath>
                  <pathelement path="${build}"/>
          </classpath>
        <batchtest fork="yes" todir="${reports}/raw/">
                  <formatter type="xml"/>
                  <fileset dir="${src}">
                          <include name="**/*Test*.java"/>
                  </fileset>
          </batchtest>
  </junit>
```

Assertion Statements

There is a list of the different types of assertion statements that are used to test your code. Any Java data type or object can be used in the statement. These assertions are taken from the JUnit API.

- **assertEquals**(expected, actual)
- **assertEquals**(message, expected, actual)
- **assertEquals**(expected, actual, delta) - used on doubles or floats, where delta is the difference in precision
- **assertEquals**(message, expected, actual, delta) - used on doubles or floats, where delta is the difference in precision
- **assertFalse**(condition)
- **assertFalse**(message, condition)
- **assertNotNull**(object)
- **assertNotNull**(message, object)

Assertion Statements (cont)

- **assertNotSame**(expected, actual)
- **assertNotSame**(message, expected, actual)
- **assertNull**(object)
- **assertNul**l(message, object)
- **assertSame**(expected, actual)
- **assertSame**(message, expected, actual)
- **assertTrue**(condition)
- **assertTrue**(message, condition)
- **fail**()
- **fail**(message)
- **failNotEquals**(message, expected, actual)
- **failNotSame**(message, expected, actual)
- **failSame**(message)

Another example

```
// A Class that adds up a string based on the ASCII values of its
// characters and then returns the binary representation of the sum.
public class BinString {
  public  BinString() {}

   public String convert(String s) {
      return binarise(sum(s));
   }

   public int sum(String s) {
      if(s=="") return 0;
      if(s.length()==1) return ((int)(s.charAt(0)));
      return ((int)(s.charAt(0)))+sum(s.substring(1));
   }

   public String binarise(int x) {
      if(x==0) return "";
      if(x%2==1) return "1"+binarise(x/2);
      return "0"+binarise(x/2);
   }
}
```


the test class

```
import junit.framework.*;
public class BinStringTest extends TestCase {
   private BinString binString;

   public BinStringTest(String name) {
      super(name);
   }

   protected void setUp() {
      binString = new BinString();
   }

   public void testSumFunction() {
      int expected = 0;
      assertEquals(expected, binString.sum(""));
      expected = 100;
      assertEquals(expected, binString.sum("d"));
      expected = 265;
      assertEquals(expected, binString.sum("Add"));
   }
```

the test class (cont)

```
public void testBinariseFunction() {
      String expected = "101";
      assertEquals(expected, binString.binarise(5));
      expected = "11111100";
      assertEquals(expected, binString.binarise(252));
   }

   public void testTotalConversion() {
     String expected = "1000001";
     assertEquals(expected, binString.convert("A"));
   }
}
```

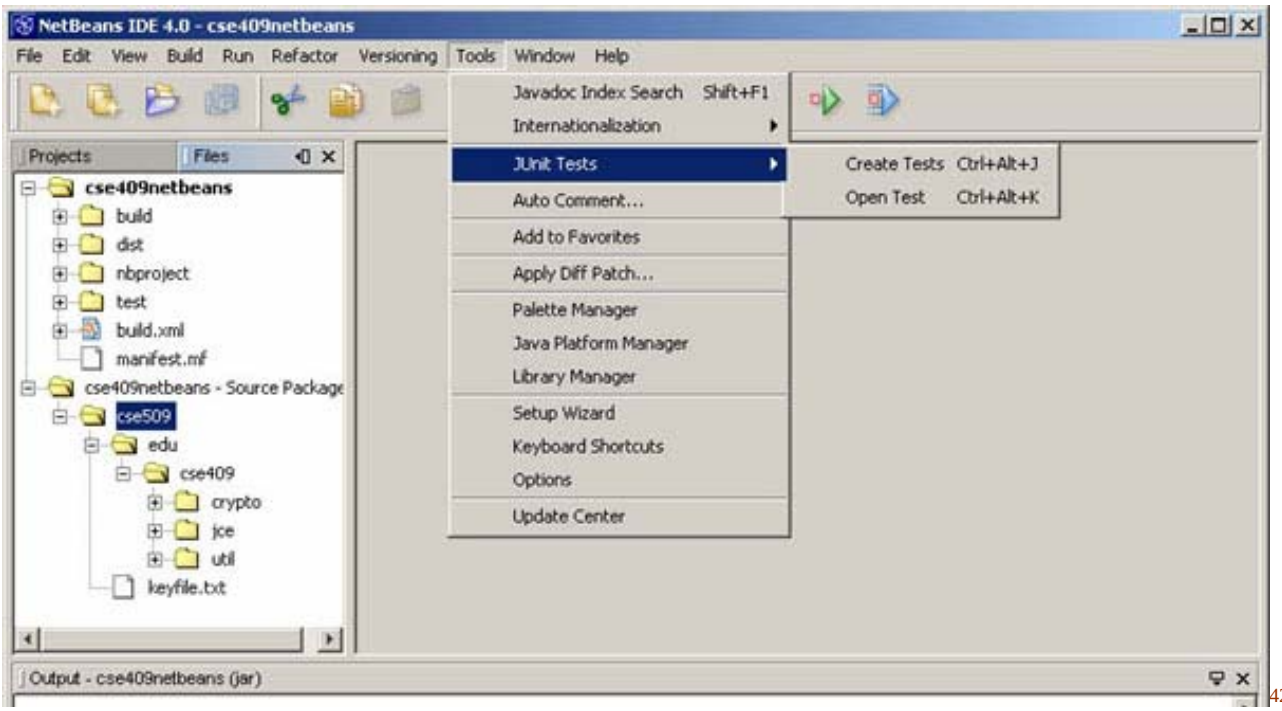

setUp and tearDown

- TestCase allows us to use the method ***setUp*** to set up any necessary variables or objects (***setUp*** is called before the evaluation of each test)
  - Note that *`binString`* was already declared at the top of the file like usual with `private BinString binString;`
- *setUp* has a brother called ***tearDown*** which is called after the evaluation of each test and can be used to dereference variables or whatever so that each test may be performed without issuing side-effects that may affect the outcome of the other tests.

Tips

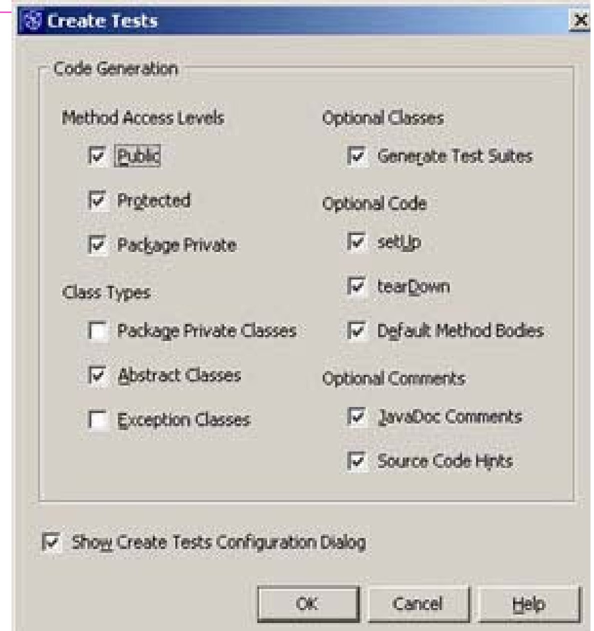
- Using an IDE (like Eclipse, NetBeans) you can create tests very quickly and easily
- Tests running can be automated

case NetBeans
creating test



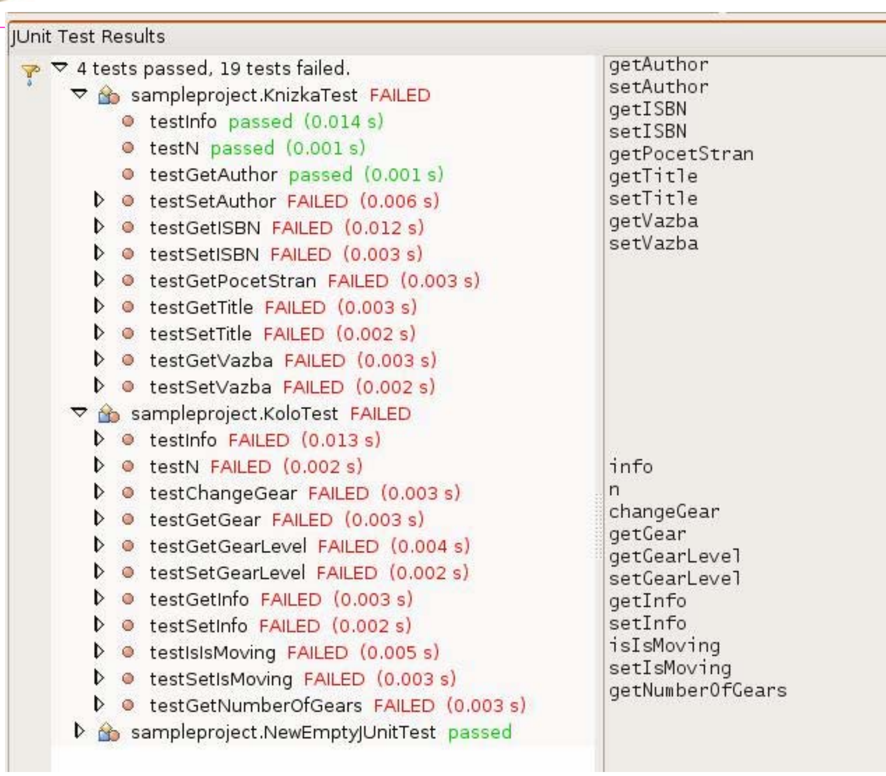
case NetBeans
selecting the elements for which tests should be created

We are now presented with a Create Tests window. You will notice several check boxes. If the methods that you are trying to test are not static meaning that they need an instance of the object to be called then you will probably want to specify a **setUp** method for the test case. This will be where you can instantiate your object before running the tests. Also if your tests allocate any special resources you may want to select **tearDown** so you can properly dispose of the resources. After we are done with this you may click the next button.



- An alternate way to create tests for individual classes is to right click on the file that you wish to create a test for then goto JUnit Tests and then Create Tests. This will allow you to create a single test for a particular class.

The results of running the tests



Documentation
(Τεκμηρίωση)

User Documentation

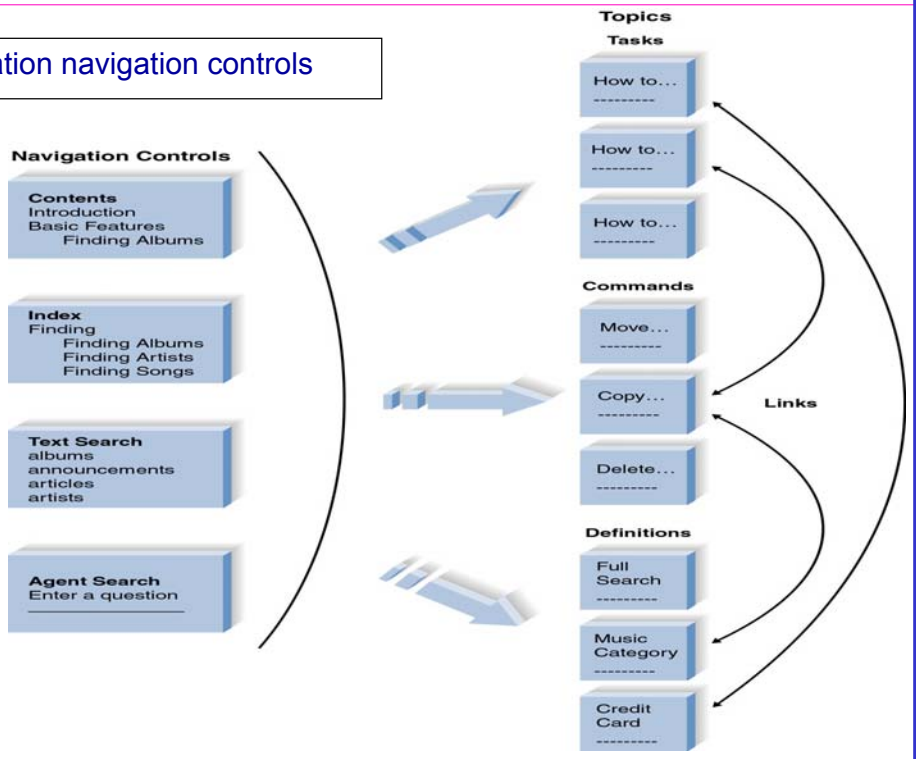
- Intended to help users operate the system
- High quality documentation takes about 3 hours per page
- The task should not be left to the end of the project
- Time required to develop and test user documentation should be built into project plan
- On-line documentation is growing in importance

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

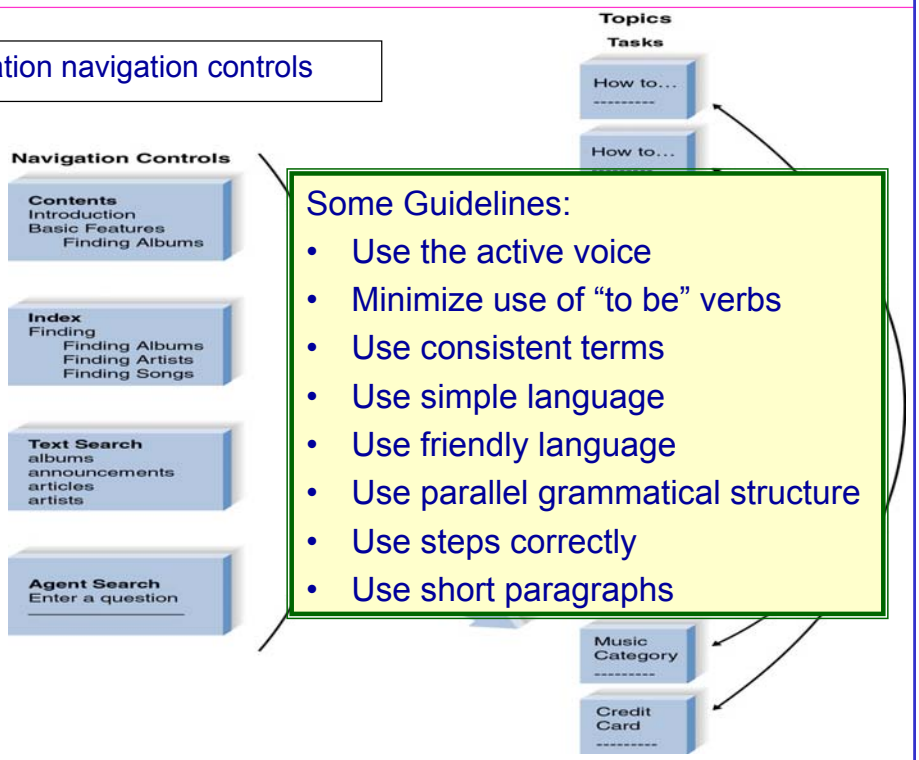
Documentation navigation controls



Documentation topics
- Commands and menus
- Common tasks
- Definitions

Organizing On-Line Reference Documents

Documentation navigation controls



Some Guidelines:
- Use the active voice
- Minimize use of “to be” verbs
- Use consistent terms
- Use simple language
- Use friendly language
- Use parallel grammatical structure
- Use steps correctly
- Use short paragraphs

Documentation topics
- Commands and menus
- Common tasks
- Definitions

Reuse !
(Επαναχρησιμοποίηση)

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.

Now we even have **search engines** for code!

- Google (http://www.google.com/codesearch)
- Krugle (http://www.krugle.org/)
- codase (http://www.codase.com/)
- SourceBank (http://archive.devx.com/sourcebank/)
- ...

Planning a Strategy for Reuse (II)

Reuse-driven Software Engineering Business

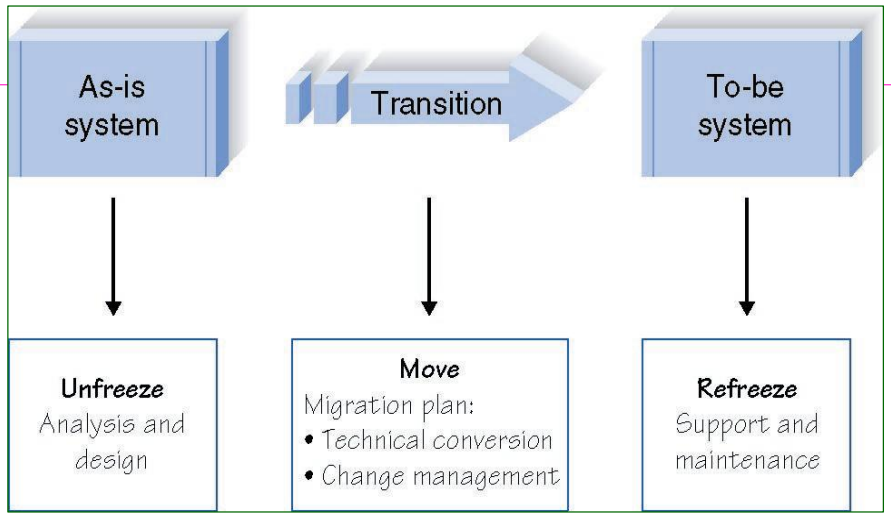
- key notions: reuse from the start
- Instead of considering components as executables (or as packages of executables designed to deliver a particular service), we can consider reuse in terms of any of the work products of systems development.
- So models used before coding are also subject to reuse: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

Component Standards

Borland Delphi	Object Pascal units compiled into .dll files - Dynamic Link Libraries
Microsoft Visual Basic	.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

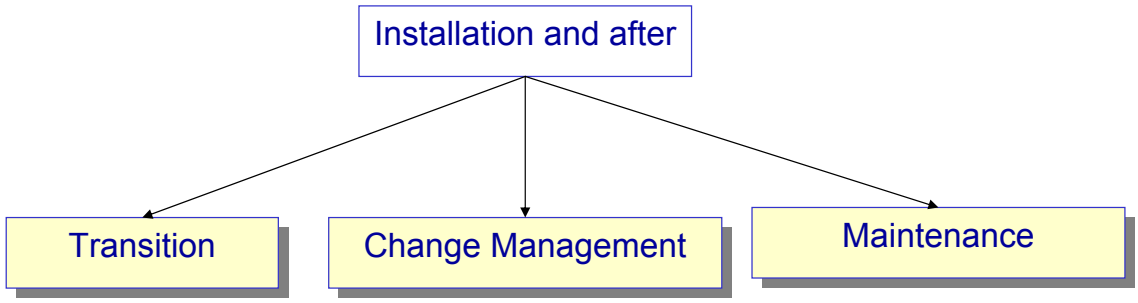
Installation and After (Εγκατάσταση και μετά)

Transition (Μετάβαση)



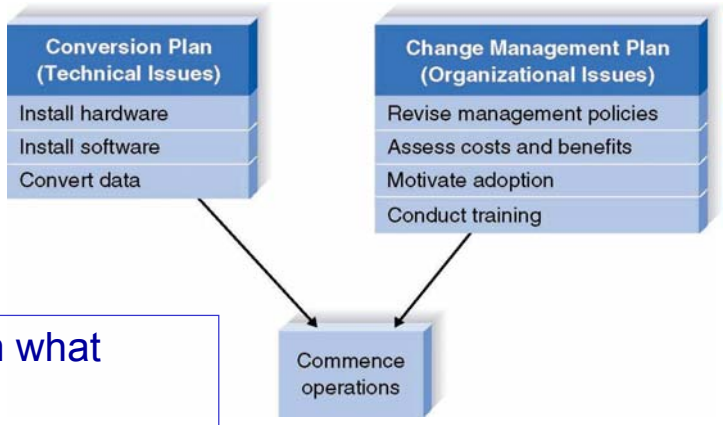
- Transitioning to new systems involves managing change from pre-existing norms and habits.
- Change management involves:
  - Unfreezing -- loosening up peoples' habits and norms
  - Moving -- transition from old to new systems
  - Refreezing -- institutionalize and make efficient the new way of doing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

Installation and After



Transition
(Μετάβαση)

Migration Plan
(Πλάνο «μετάβασης»)



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

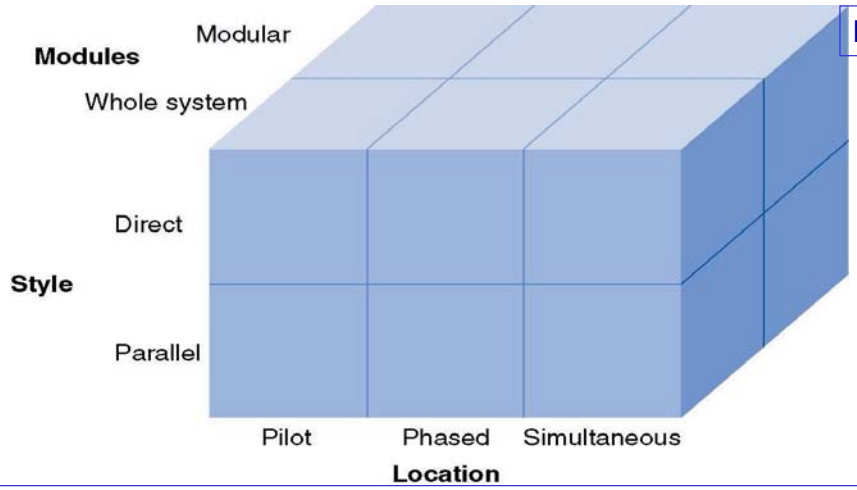
Migration Plan

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 (Στρατηγικές Μετάβασης)



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 (Διαχείριση Αλλαγής)

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
(Εκπαίδευση)

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
(«Εγκαθίδρυση» του Συστήματος)

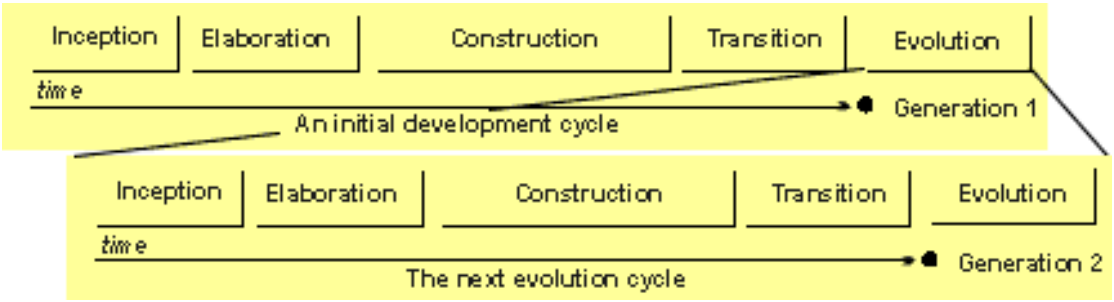
Types of System Support

- 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 (Δραστηριότητες Μετά την Υλοποίηση)

- 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

System Support and **Maintenance** using CASE tools

Case Study: Enterprise Architect

EA: Maintenance Model

Aim

- The **Maintenance Model** captures issues, bugs, tests and other information related to maintaining and updating the software system. This model is typically created once the first release of the product is shipped, and is used to ensure all bugs and problems are stored and handled in the correct way.

Diagram

- The `Maintenance Model` allows visual representation of issues arising during and after development of a software product.. The Model can be enhanced with the integration of change elements and testing.

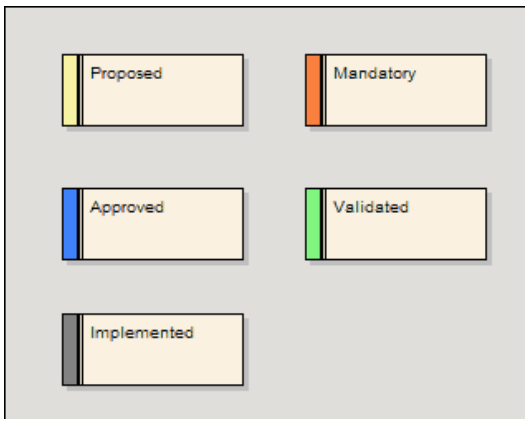
EA: Maintenance Elements

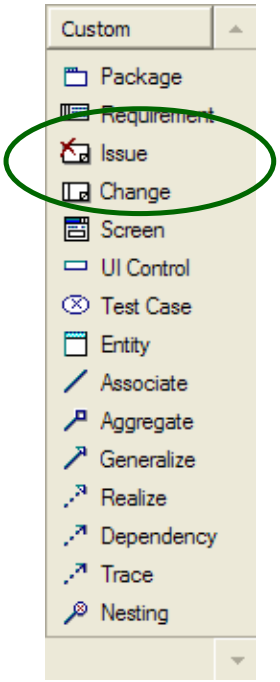
The maintenance elements are **defects**, **changes**, **issues** and **tasks**. They all apply to individual model elements and may be used to record and capture problems, changes, issues and tasks as they arise and document the solution and associated details. They are defined as follows:

- A **defect** can be considered as a failure to meet a requirement for the current model element.·
- A **change** can be considered as a change in requirement for the current model element.
- An **issue** is a record of a risk or other factor that might affect the project being recorded for the current model element.
- A **task** is a means of recording work in progress and work outstanding for the current model element.

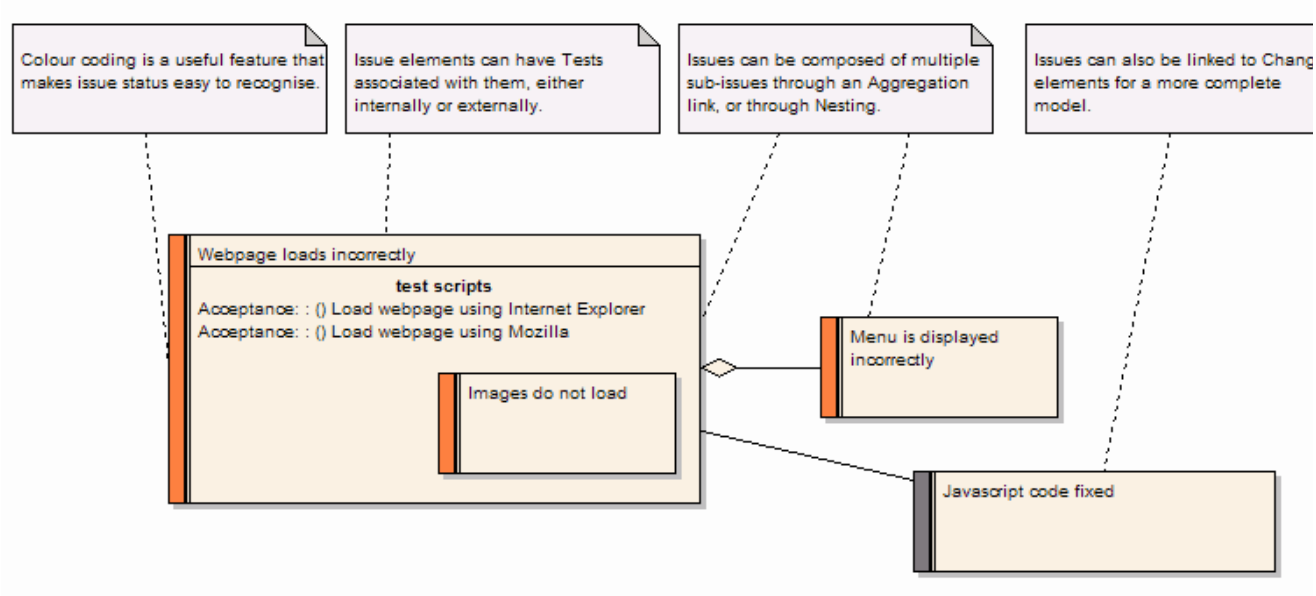
Note that each of these maintenance elements applies at the model element level.

EA: Coloring scheme and Diagrammatic Elements

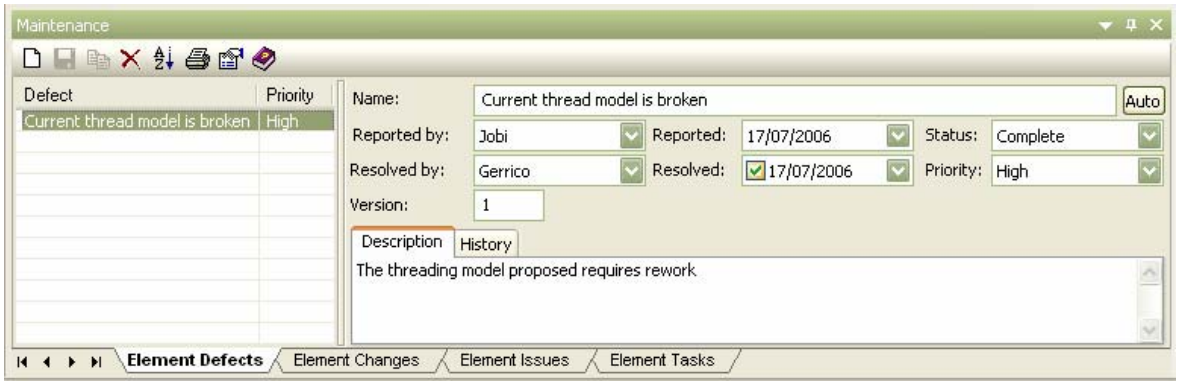




EA: Maintenance Model pattern



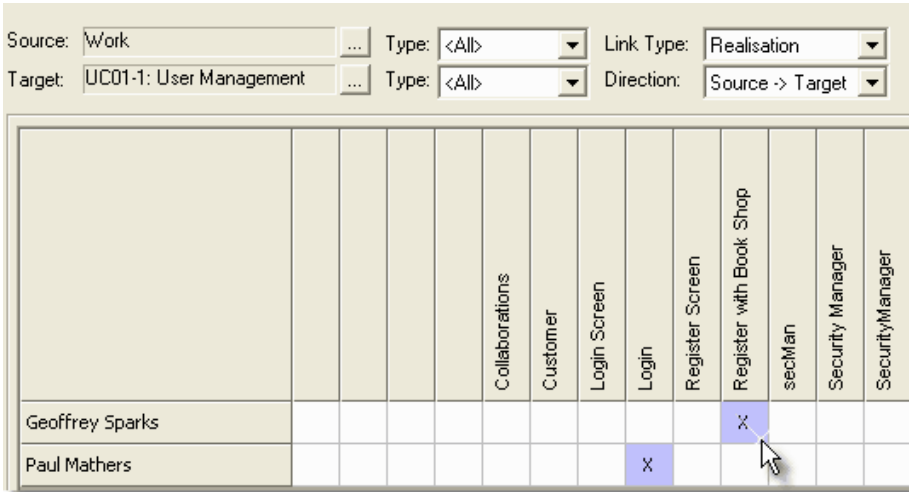
EA: Maintenance Workspace



- EA makes it easy to record and capture problems and issues as they arise and document the solution and associated details. The Maintenance window provides a quick method of viewing and modifying the list of defects, changes, issues and to do items associated with a particular model element. Access this window by selecting Maintenance from the View menu, or by pressing Alt+4.
- Four tabs provide access to Element Defects, Element Changes, Element Issues and Element Tasks - click on the tab of interest then select model elements in diagrams or in the Project Browser to see the associated maintenance items. You can include defects, changes, issues and tasks in the main RTF documentation and HTML produced by EA. The RTF setup dialog has check boxes to show or hide element defects, changes, issues and tasks

EA: Assign People to Defects or Changes

- As an example of how you might use the relationship matrix to monitor issues or changes - the screen below illustrates staff (actors) being linked through Realization connectors to Issues. Each highlighted square indicates a responsibility that a staff member has to work on or correct a named issue.
- This same approach can be used for any mix of model elements.



Project Assessment (Αποτίμηση Έργου)

- 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)