UCSI University

Topic 1
1.1 Implementation

Based on Chapter 19 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML.

Overview

- About tools used in software implementation
- How to draw component diagrams?
- How to draw deployment diagrams?
- The tasks involved in testing a system
- How to plan for data conversion from an old system?
- Ways of introducing a new system into an organization
- Tasks in review and maintenance

Software Implementation Tools

- In an iterative project, planning for implementation will begin in the inception phase.
- The implementation workflow includes tasks to set up the environment for implementation.
- Some tools, particularly modelling tools and configuration management systems will carry through from analysis and design activities.
- A wide range of types of software tools will be used.

Software Implementation Tools

- Modelling Tools
  - Many tools now support UML.
  - Make it possible to generate code (in Java, C++ and VB) from the models.
  - May make reverse engineering of code possible, to provide round-trip engineering.
  - May map classes to a relational database.
  - Link to configuration management tools.

Software Implementation Tools

- Compilers, Interpreters and Run-times
  - Different languages require different tools.
  - C++ requires a compiler and a linker to build executables.
  - Java requires a compiler and a run-time program and libraries to run the byte-code produced by the compiler.
  - C# is like Java and is compiled into MSIL (Microsoft Intermediate Language).

Software Implementation Tools

- Visual Editors
  - Provide a way of designing GUI interfaces by dragging and dropping buttons, text fields etc. onto a window.
  - Can often also handle controls or objects that represent non-visual components such as links to a database or communications processes.

Software Implementation Tools

- IDEs (Integrated Development Environments)
  - Manage the many files in a project and the dependencies among them.
  - Link to configuration management tools.
  - Use compilers to build the project, only recompiling what has changed.
  - Provide debugging facilities.
  - May include a visual editor.
  - Can be configured to link in third party tools.

Software Implementation Tools

- Configuration Management Tools
  - Also called version control tools, although configuration management is more than just version control.
  - Maintain a record of file versions and the changes from one version to the next.
  - Record all the versions of software and tools that are required to produce a repeatable software build.

Software Implementation Tools

- Class Browsers
  - May be part of IDEs or visual editors.
  - Originally provided as the way of browsing through available classes in Smalltalk.
  - Java API documentation is provided in a browsable hypertext format generated by Javadoc.

Software Implementation Tools

- Component Managers
  - Tool to manage components and their dependencies.
  - Provide mechanisms to do the following:
    - Add components
    - Search components
    - Browse components
    - Maintain versions of components

Software Implementation Tools

- DBMS (Database Management Systems)
  - Server system
  - Client software (administration interfaces, ODBC and JDBC drivers)
  - Tools to manage the database and carry out performance tuning.
  - Large DBMS, such as Oracle, come with many tools, even their own application server.

Software Implementation Tools

- CORBA
  - CORBA ORB to handle the marshalling and unmarshalling of requests and objects.
  - IDL compiler
  - Registry service

Software Implementation Tools

- Application Containers
  - Web containers
    - Like Tomcat
    - Run servlets and small-scale applications
  - Application servers
    - Like WebSphere, WebLogic or Jboss
    - Provide a framework to run large-scale, enterprise applications.

Software Implementation Tools

- Testing Tools
  - Tools written by developers as test harnesses.
  - Automated test tools to run repeated or multiple simultaneous tests.
  - May allow user to run through test once manually, then generate a script that can be edited to provide variations.

Software Implementation Tools

- Installation Tools
  - Automate the extraction of files from an archive and the setting up of configuration files and registry entries.
  - Some maintain information about dependencies on other pieces of software and will install all necessary packages (e.g. Redhat RPM).
  - Uninstall software, removing files, directories and registry entries.

Software Implementation Tools

- Conversion Tools
  - Extract data from existing systems.
  - Reformat the data for the new system.
  - Insert it into the database for the new system.
  - May require manual intervention to 'clean up' the data such as removing duplication or invalid values in fields.

Software Implementation Tools

- Documentation Generators
  - Document models and code
  - Extract standard information or user-defined information into document templates.
  - Produce HTML to document the API of classes in the application.

Exercise

- Based on the implementation tools discussed previously, name the latest tools that we can use for software implementation.

Coding and Documentation Standards

- Naming standards are agreed early in a project.
- A typical object-oriented standard:
  - classes with capital letters: **`Campaign`**
  - attributes and operations with initial lower case letters: **`title`**, **`recordPayment()`**
  - words are concatenated together with capital letters to show where they are joined: **`InternationalCampaign`**, **`campaignFinishDate`**, **`getNotes()`**

Coding and Documentation Standards

UCSI University

- Hungarian Notation
- Used in C and C++
- Names prefixed by an abbreviation to show the type of the member variable
  - b for boolean: **`bOrderClosed`**
  - i for integer: **`iOrderLineNumber`**
  - btn for button: **`btnCloseOrder`**

Coding and Documentation Standards

UCSI University

- One other standard:
  - using underscores to separate parts of a name instead of capital letters.
  - `Order_Closed`
  - often used for column names in databases, as it is easier to replace the underscores with spaces to produce meaningful column headings in reports than trying to find the word breaks in concatenated names.

Coding and Documentation Standards

UCSI University

- Document code
  - Think of the people who will maintain your code.
  - Others may be able to use your code to learn good practice, but only if it is clearly documented.
  - No language is self-documenting; conventions and standards help.
  - Comply with Java documentation standards, if coding in Java (Javadoc).
  - Use XML tags if coding in C#.
  - You can take advantage of tools that automate the production of documentation from comments.

Implementation Diagrams

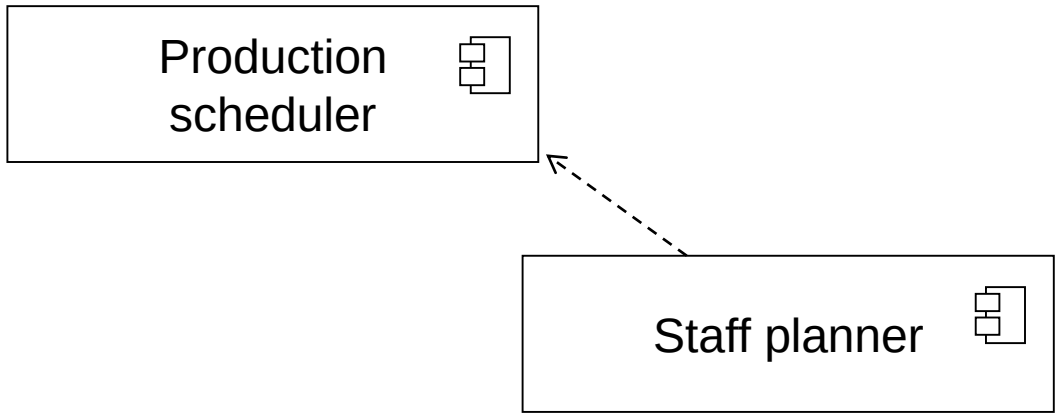
- Component Diagrams
  - used to document dependencies between components, which are modular software units with a well-defined interface.
- Deployment Diagrams
  - used to show the configuration of run-time processing elements and the software artefacts and processes that are located on them.

Notation of Component Diagrams

- Components
  - Rectangles with a component icon  in the top right-hand corner.
  - May provide or require interfaces.
- Dependencies

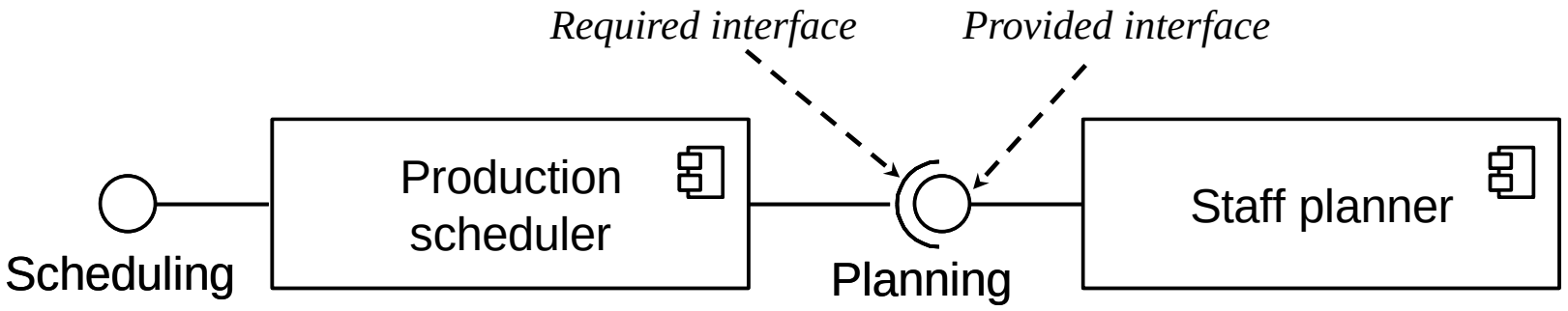
Notation of Component Diagrams

- Dependencies between components



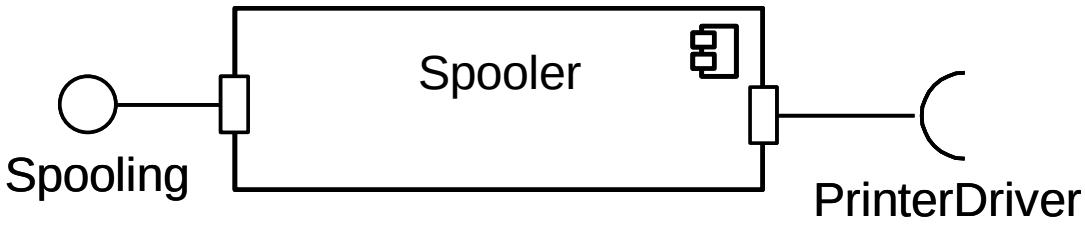
Notation of Component Diagrams

- Wiring connection between required and provided interfaces.



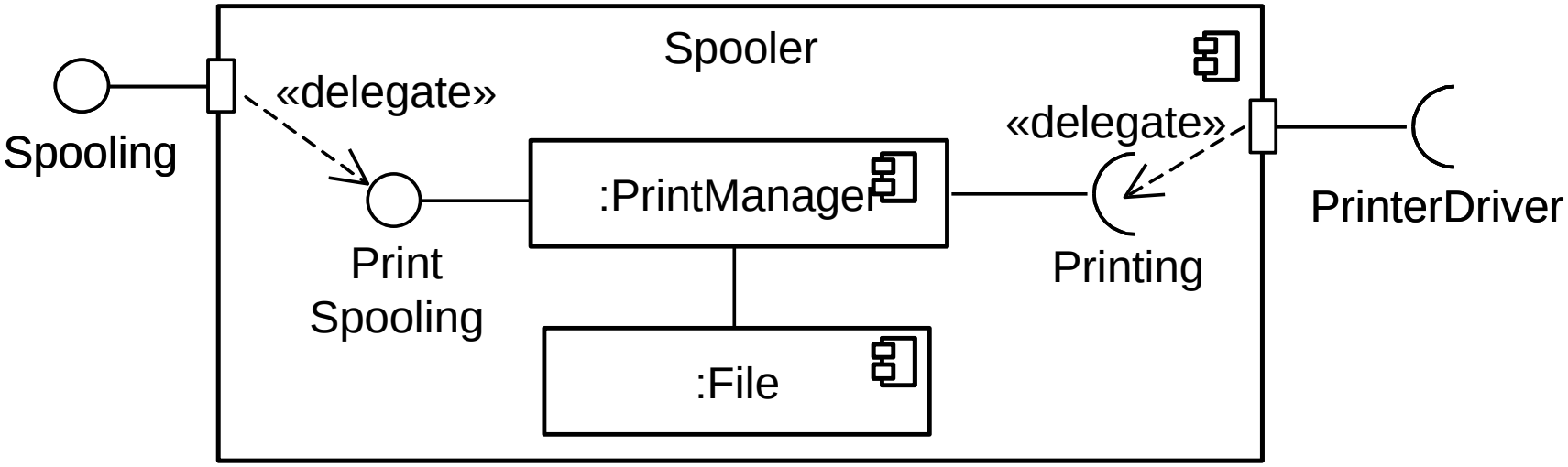
Notation of Component Diagrams

- Component with ports
  - Indicates that the component delegates responsibility for the behaviour of that interface to a subcomponent.



Notation of Component Diagrams

- Component with ports
  - Shows delegated responsibility



Components

- Components have changed in UML 2.0
- They are no longer shown in Deployment Diagrams, where they have been replaced by Artefacts.
- Components are specifically modular software units with well-defined interfaces.
- They can be logical or physical.

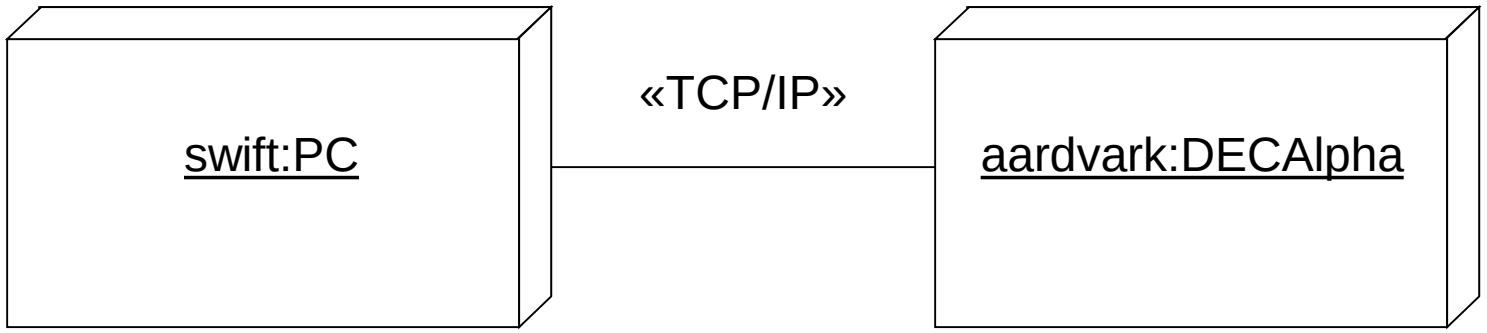
Forms of Component

- Cheesman and Daniels make a clear distinction between different types of component:
  - Component specification
  - Component implementation
  - Installed component
  - Component object

Notation of Deployment Diagrams

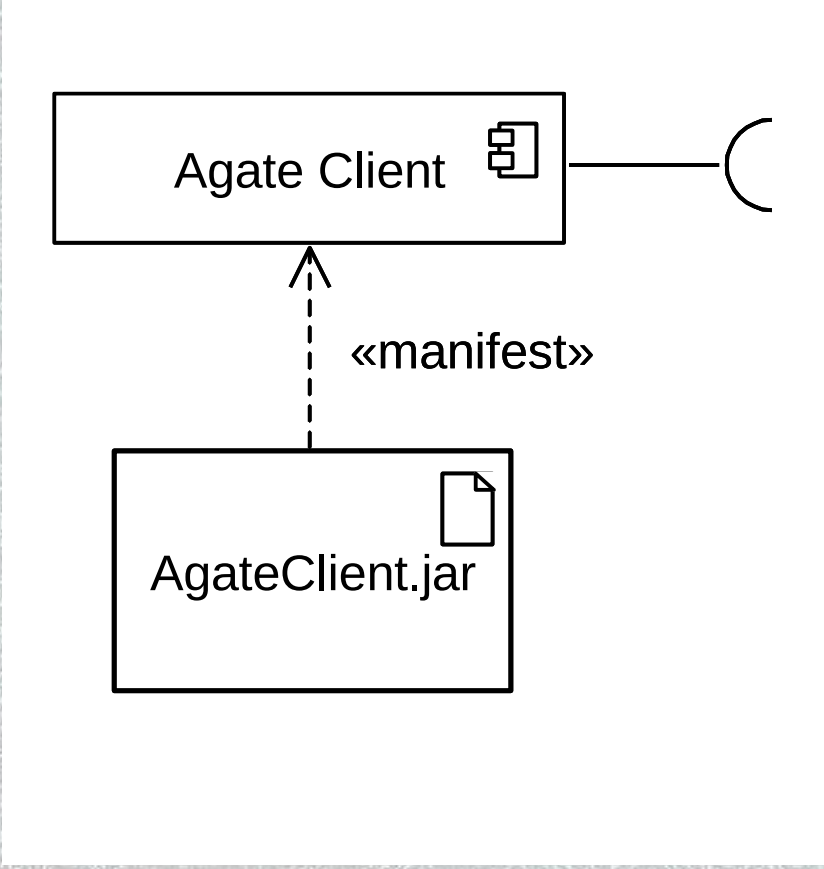
- Nodes
  - Rectangular prisms
  - Represent processors, devices or other resources
- Communication Associations
  - Lines between nodes
  - Represent communication between nodes
  - Can be stereotyped

Notation of Deployment Diagrams

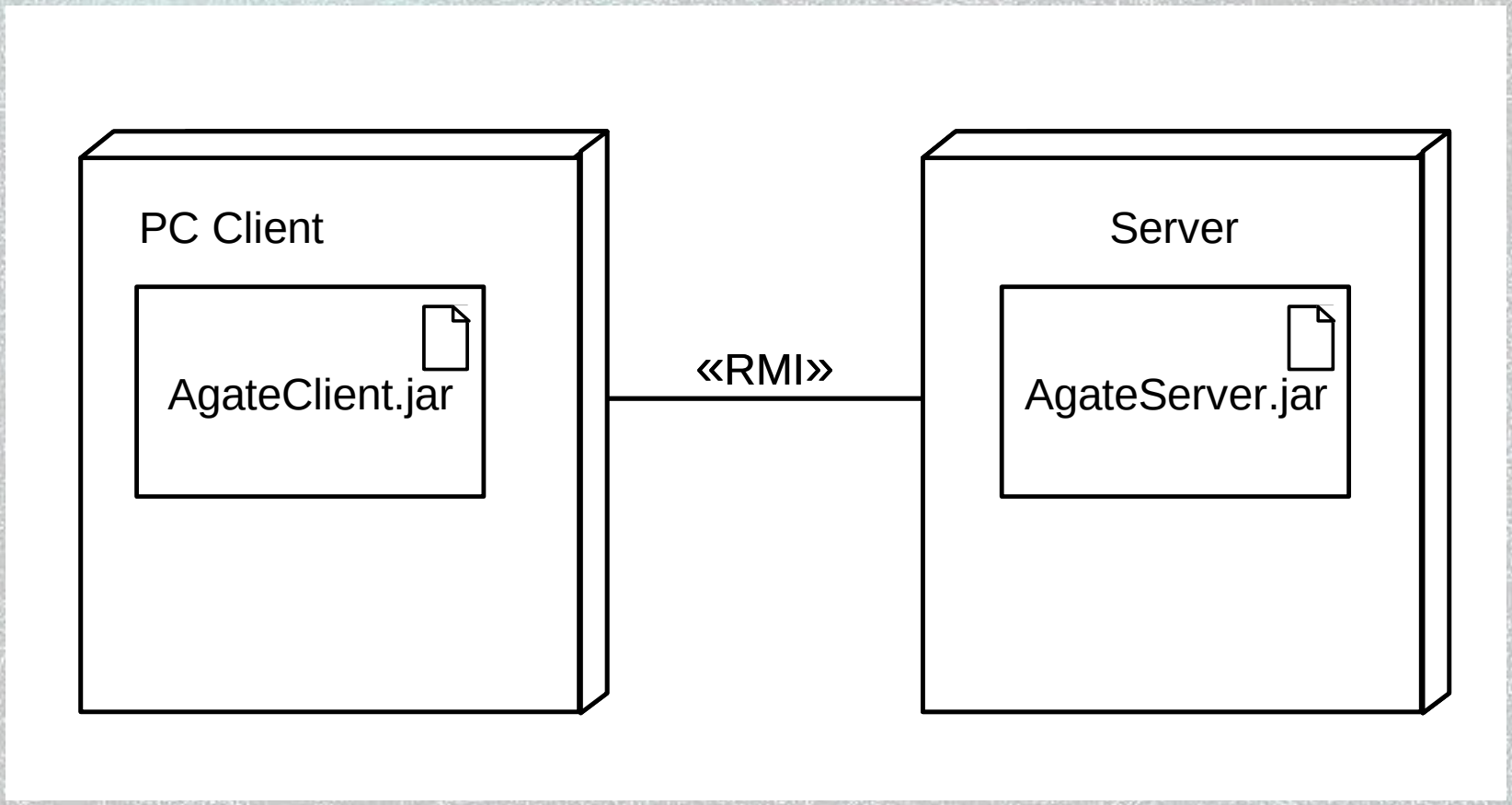


Notation of Deployment Diagrams

- Can be shown with artefacts located on the nodes.
- Artefacts are related to components by a 'manifest' relationship.

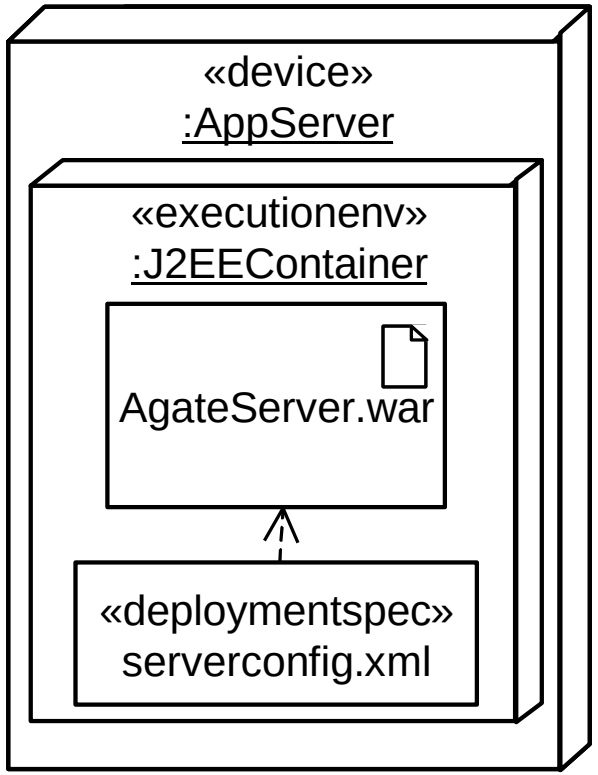


Notation of Deployment Diagrams



Notation of Deployment Diagrams

- Nodes can be stereotyped.
- Execution environments represent application containers.
- Deployment specifications describe the configuration of artefacts.



Use of Implementation Diagrams

- Can be used architecturally to show how elements of system will work together.
- Typically used for simple diagrams.
- Full documentation of dependencies and location of all components may be better handled by configuration management software or in a spreadsheet or database.

Data Conversion

- Data from manual systems needs collating and putting into a standard format.
- Incomplete data may need to be chased up.
- There is a cost associated with keying data into a new system.
- There may be a requirement for data entry screens that will only be used to get data into the system to start it up.

Data Conversion

- Data from existing computer systems
  - Existing data must be checked for correctness.
  - Specially written programs may be required to check and convert the data.
  - Data may be loaded into a staging area to be 'cleaned up'.
  - Data is imported into the new system.
  - Data must be verified after being imported.

User Documentation and Training

- User manuals
  - Training manuals organized around the tasks the users carry out.
  - On-line computer-based training (CBT) that can be delivered when the users need it.
  - Reference manuals to provide complete description of the system in terms the users can understand.
  - On-line help replicating the manuals.

User Documentation and Training

UCSI University

- User Training
  - Set clear learning objectives for trainees.
  - Training should be practical and geared to the tasks the users will carry out.
  - Training should be delivered 'just in time' not weeks before the users need it.
  - CBT can deliver 'just in time' training.
  - Follow up after the introduction of the system to make sure users haven't got into bad habits through lack of training or having forgotten what they had been told.

Review

- Post-implementation review
- Review the system
  - whether it is delivering the benefits expected
  - whether it meets the requirements
- Review the development project
  - Record lessons learned
  - Use actual time spent on project to improve estimating process
- Plan actions for any maintenance or enhancements.

Evaluation Report

- Cost benefit analysis—Has it delivered? Compare actuals with projections.
- Functional requirements—Have they been met? Any further work needed?
- Non-functional requirements—Assess whether measurable objectives have been met.
- User satisfaction—Quantitative and qualitative assessments of satisfaction with the product.
- Problems and issues—Problems during the project and solutions so lessons can be learned.

Evaluation Report

- Positive experiences—What went well? Who deserves credit?
- Quantitative data for planning—How close were time estimates to actuals? How can we use this data?
- Candidate components for reuse—Are there components that could be reused in other projects in the future?
- Future developments—Were requirements left out of the project due to time pressure? When should they be developed?
- Actions—Summary list of actions, responsibilities and deadlines

Maintenance Activities

- Systems need maintaining after they have gone live.
- Bugs will appear and need fixing.
- Enhancements to the system may be requested.
- Maintenance needs to be controlled so that bugs are not introduced and unnecessary changes are not made.

Maintenance Activities

- Helpdesk, operations and support staff need training to take on these tasks.
- A Change Control System is required to manage requests for bug fixes and enhancements.
- Changes need to be evaluated for their cost and their impact on other parts of the system, and then planned.

Maintenance Documentation

- Bug reporting database
- Requests for enhancements
- Feedback to users
- Implementation plans for changes
- Updated technical and user documentation
- Records of changes made

Summary

- Tools used in software implementation
- How to draw component diagrams
- How to draw deployment diagrams
- The tasks involved in testing a system
- How to plan for data conversion
- Ways of introducing a new system into an organization
- Tasks in review and maintenance