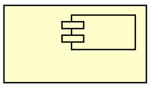
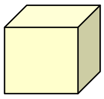
ΗΥ 351: Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
CS 351: Information Systems Analysis and Design

Physical (or Implementation) Diagrams

- UML component diagrams 
- UML deployment diagrams 

Lecture : (18b) or 19
Date : 20-12-2005

Yannis Tzitzikas
University of Crete, Fall 2005

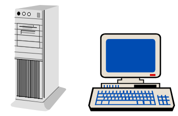
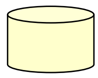
Outline

- Component Diagrams
- Deployment Diagrams
- Combining Component & Deployment Diagrams

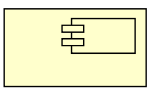
Key questions

Computing platform comprises hardware, software (PLs, DBMSs) and networking.

- *Which platform is best suited for this information system?*
- *How to select hardware ?*
- *How to select software ?*
- *How to select networking?*
- *How to express the physical architecture (see Lecture 18) of the system using a standard diagrammatic notation?*

Component Diagrams

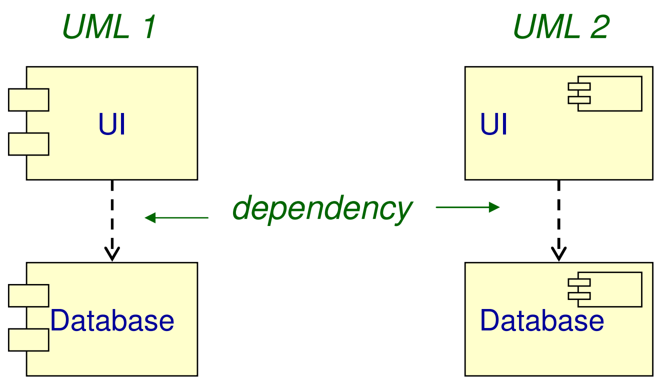


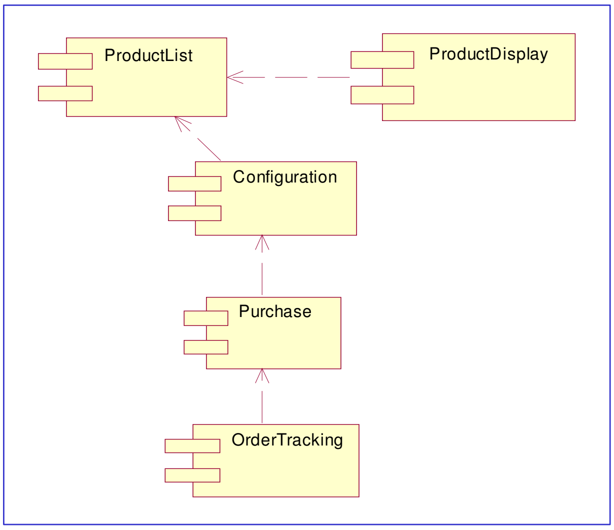
Component Diagrams (διαγράμματα εξαρτημάτων)

Component Diagrams show various <u>components</u> and their <u>dependencies</u>

- **Component**:
  - <u>physical module of code</u> (like package, class, or even file)
- **dependence**:
  - <u>change</u> dependency (e.g. communication dependencies, compilation dependencies)

Notations :





The Characteristics of a Component

- a unit of independent deployment (<u>never deployed partially</u>)
- sufficiently documented and <u>self-contained</u> to be "plugged into" other components by a third-party
- it cannot be distinguished from copies of its own; in any given application, there will be at most one copy of a particular component
- it is a <u>replaceable part of a system</u> (can be replaced by another component that conforms to the same interface)
- it fulfils a <u>clear function</u> and is logically and physically cohesive
- it <u>may be nested</u> in other components

[Szyperski 98, Rumbaugh et al. 99, Maciaszek 2005)]

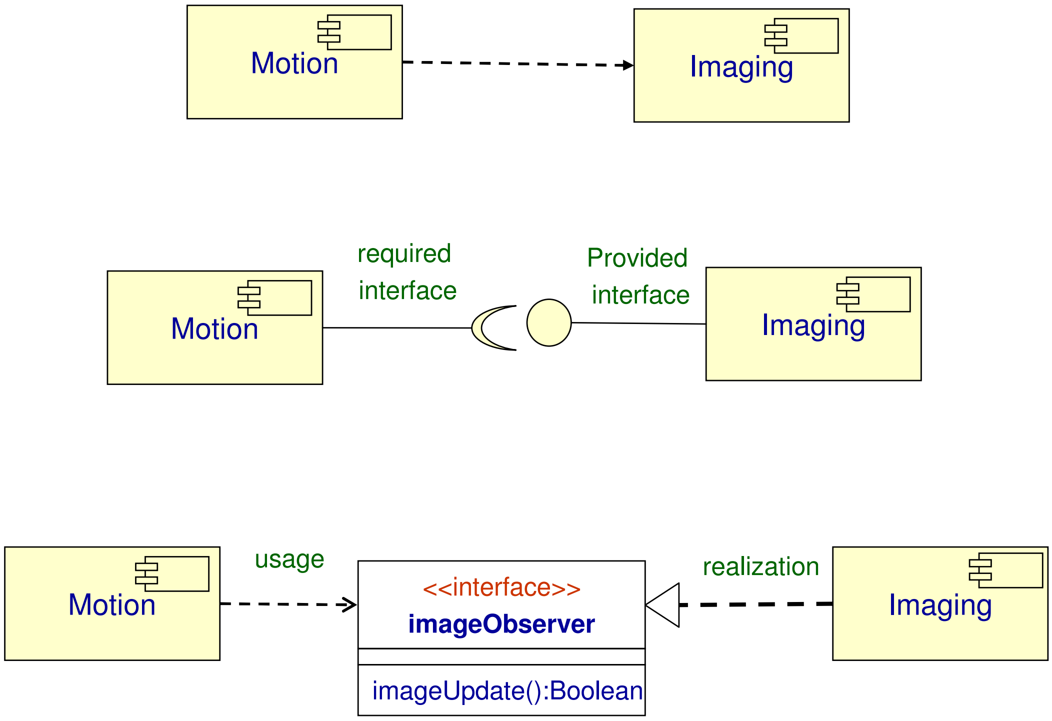
Components

- Components are like classes and packages
  - can be connected through interfaces
- Components are about how customers want to relate to software
  - they want to be able to upgrade it like they can upgrade their stereo (in pieces)
  - they want to mix and match pieces from various manufacturers
    - reasonable but difficult to satisfy
- So we could define a component as:
  - a logical and replaceable part of a system that conforms to and provides the realization of a set of interfaces
  - an independently purchasable and upgradeable piece of software

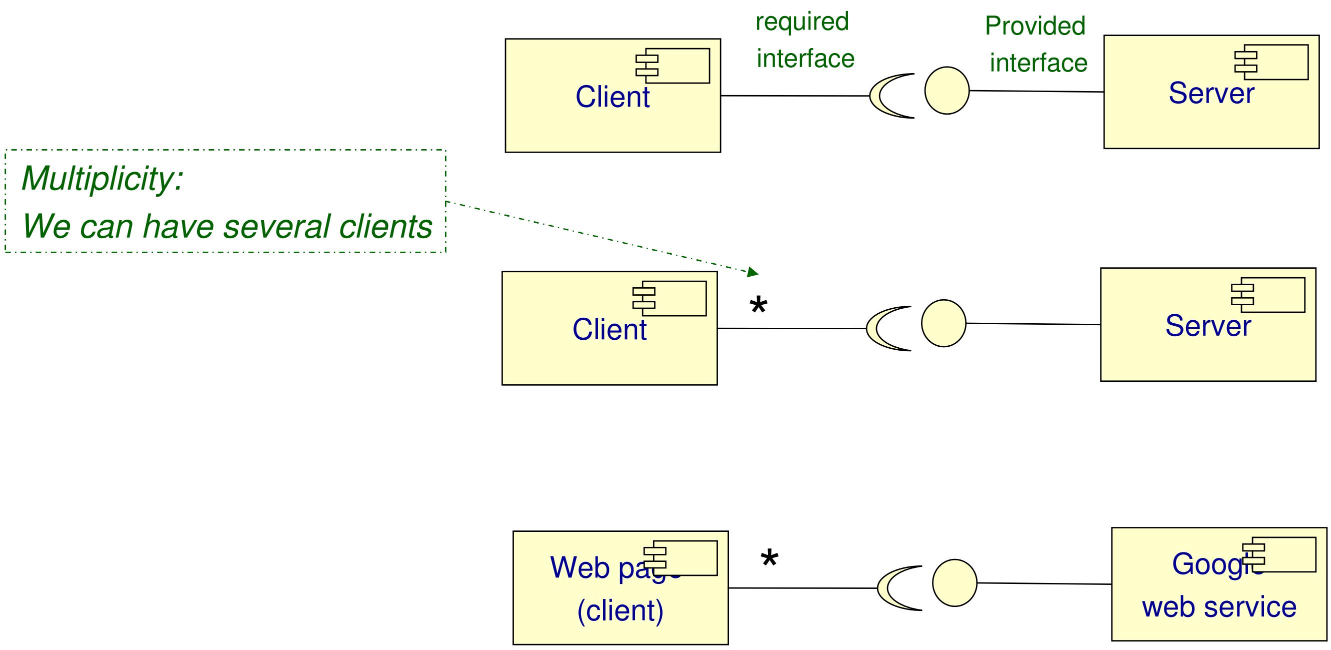
Components and related Notions

- Component
  - a replaceable part of a system that conforms to and provides the realization of a set of interfaces
- Interface:
  - a collection of operations that specify a service that is provided by or requested from a class or component
- Port
  - a specific window into an encapsulated component accepting messages to and from the component conforming to specified interfaces
- Part
  - (an internal component) the specification of a role that composes part of the implementation of a component.
- Internal structure
  - the implementation of a component by means of a set of parts that are connected together in a specific way
- Connector:
  - a communication relationship between two parts or ports within the context of component

Components and interfaces

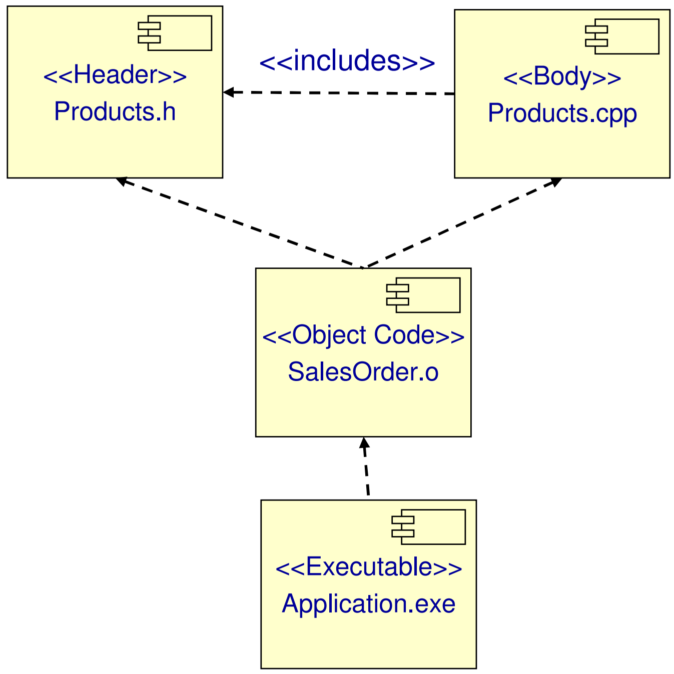


Components and interfaces (II)

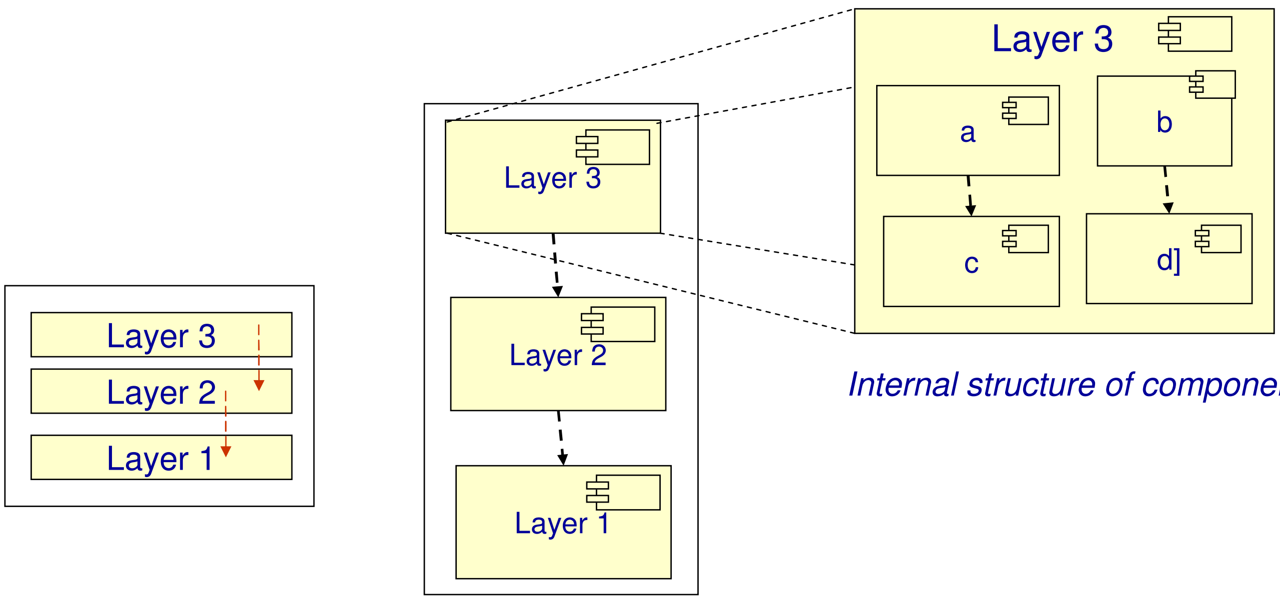


Fine-grained Components: Example

We could use component diagrams for modeling more fine-grained components (e.g. files).

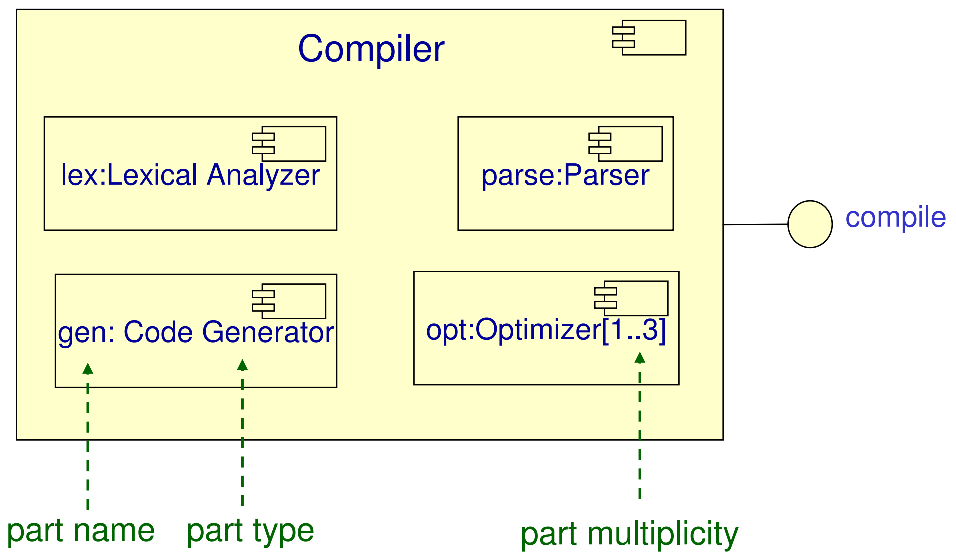


Coarse-grained components: e.g. Layers



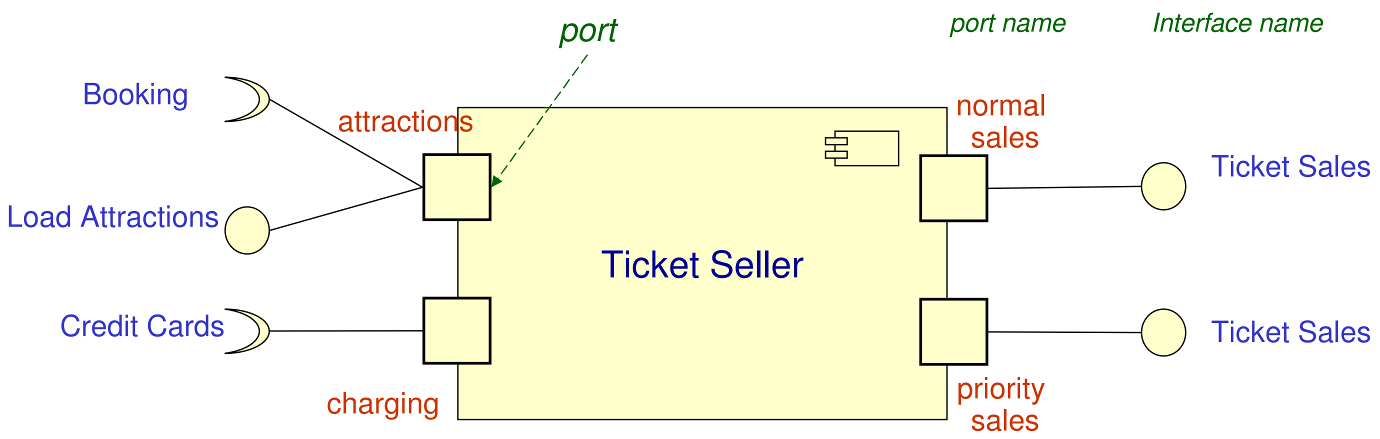
Internal structure of component

Internal Structure of Components



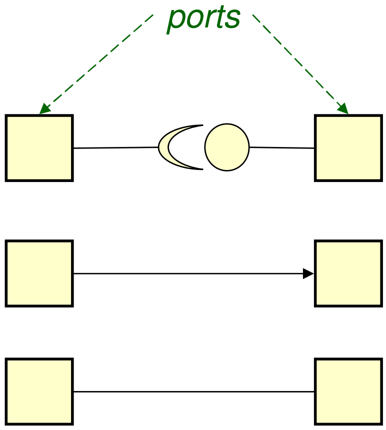
Ports

- Ports permit the interfaces of a component to be divided into discrete packets and used independently
- The externally visible behaviour of the component is the sum of its ports.



Connecting Components

- Components can be connected by wiring together their **ports**
  - **connector**: a wire between two ports

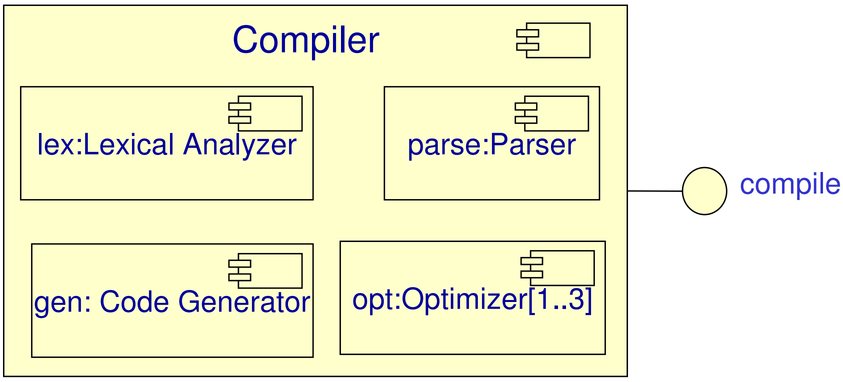


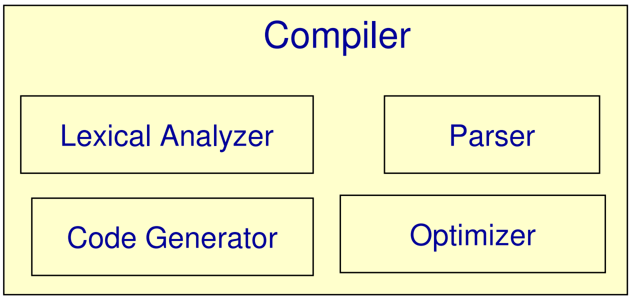
connector by interfaces

delegation connector
(connect an external port with the port of a part component)

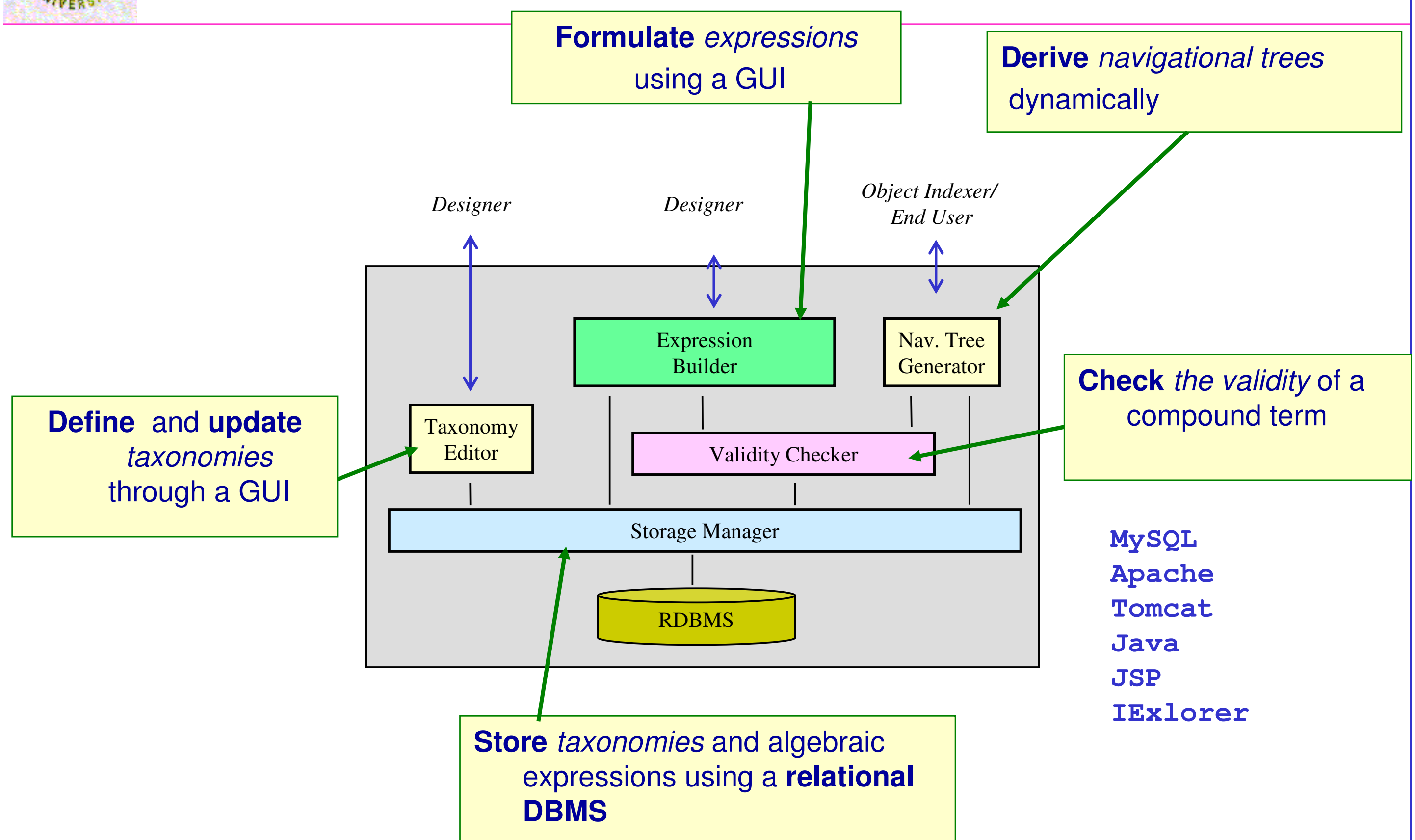
direct connector
(more tight coupling)

In practice, components diagrams are sometimes depicted in a less formal and more liberal graphical notation



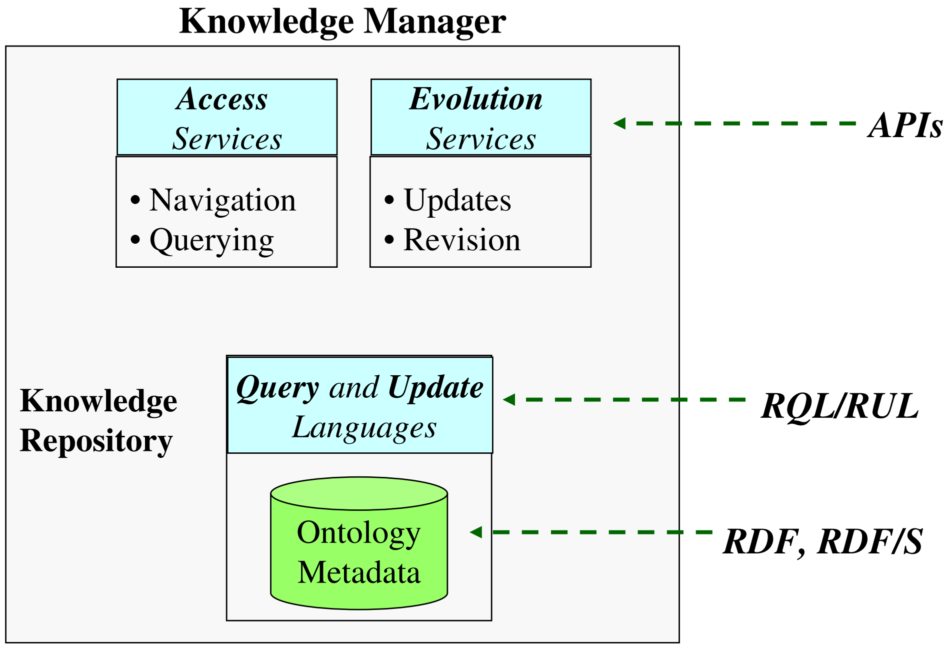


FASTAXON (functional) architecture

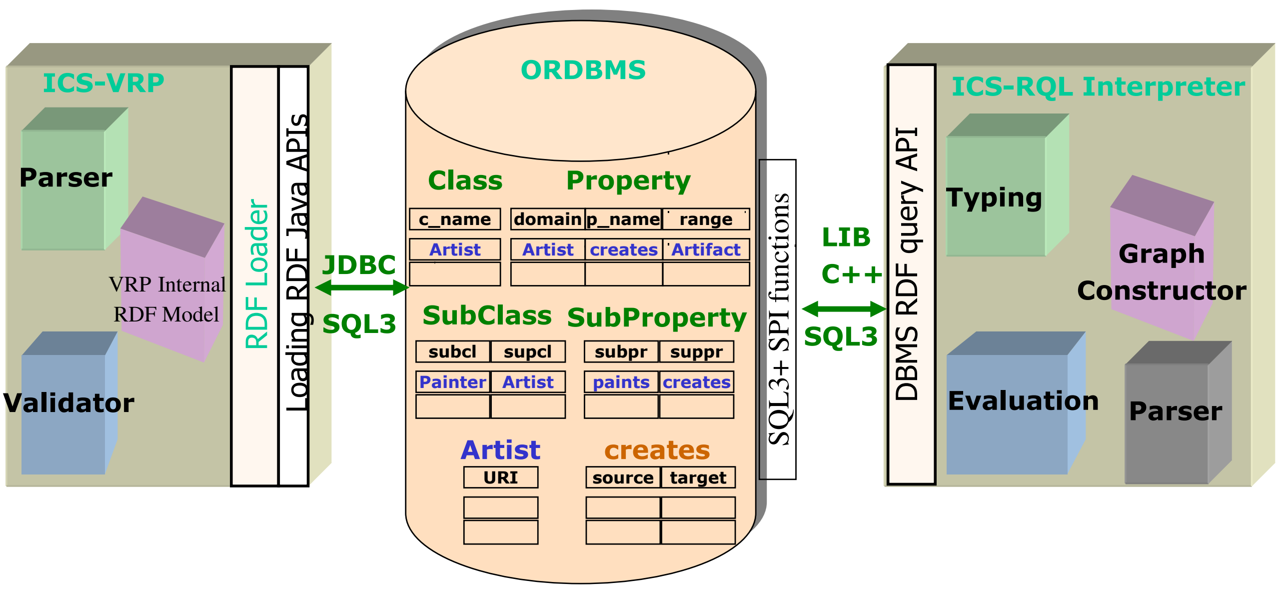


MySQL
Apache
Tomcat
Java
JSP
IExlorer

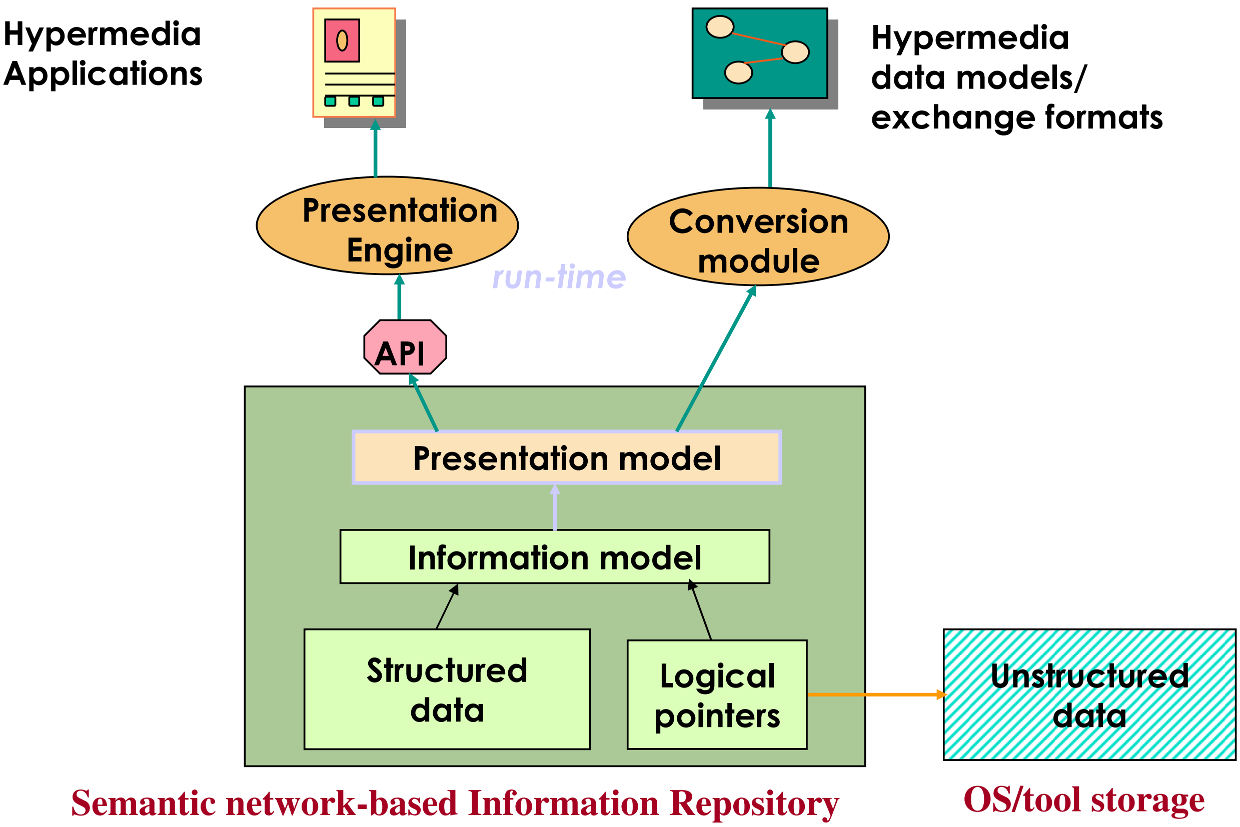
Knowledge Manager



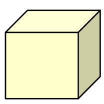
RDF Suite Architecture



DOMENICUS Architecture



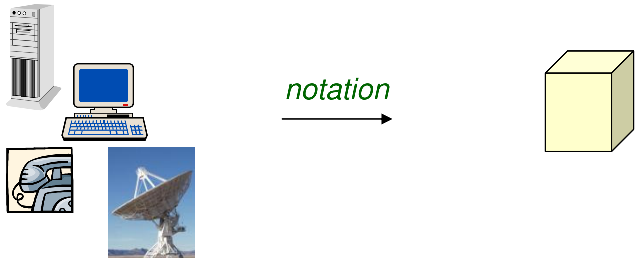
UML Deployment Diagrams



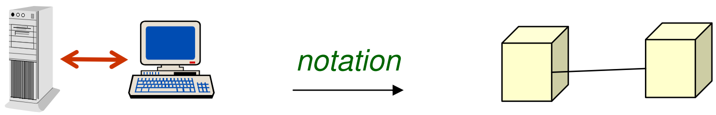
Deployment Diagrams
(διαγράμματα ανάπτυξης/σύνταξης/παράθεσης)

Shows the physical relationship among software & hardware components in the delivered system

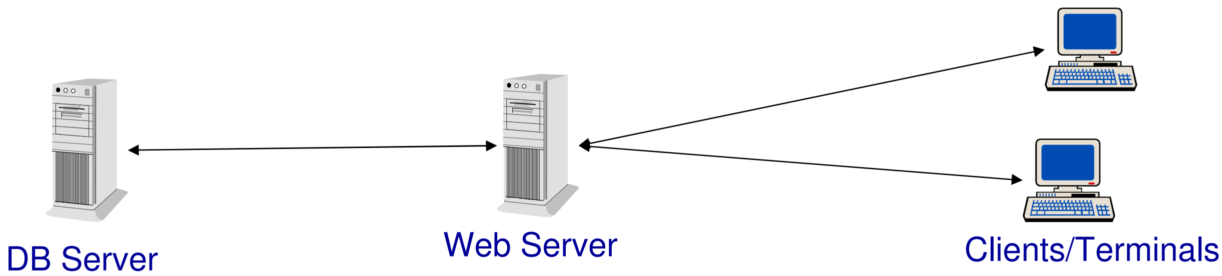
Node:
- computational unit (hardware)
  - e.g. PC, sensor, mainframe, mobile device

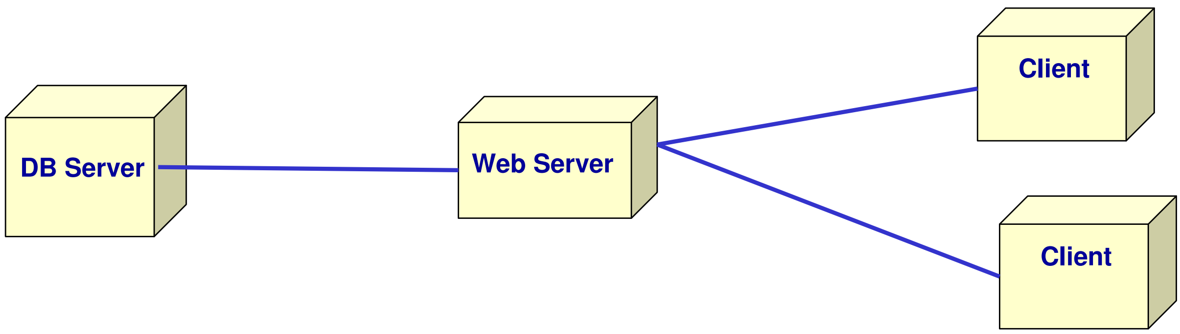


Connection (among nodes)
- communication paths over which the system will interact

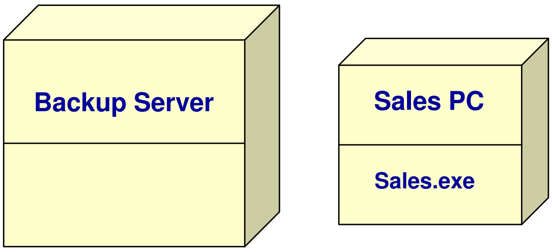


A deployment diagram

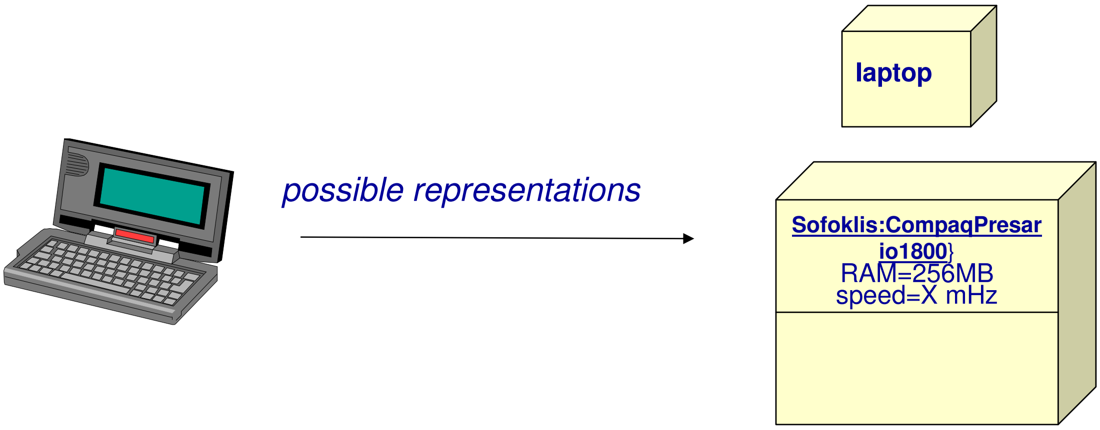




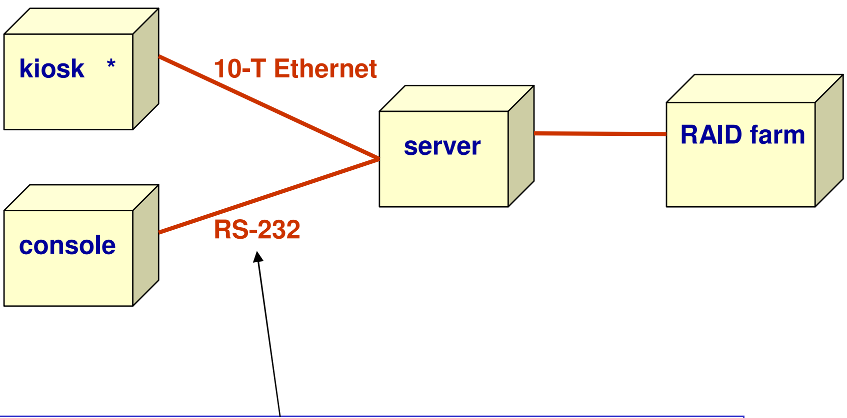
Deployment Diagrams> Nodes



- Physical element (with memory and processor)
- With nodes we can model the topology of the hardware of a system



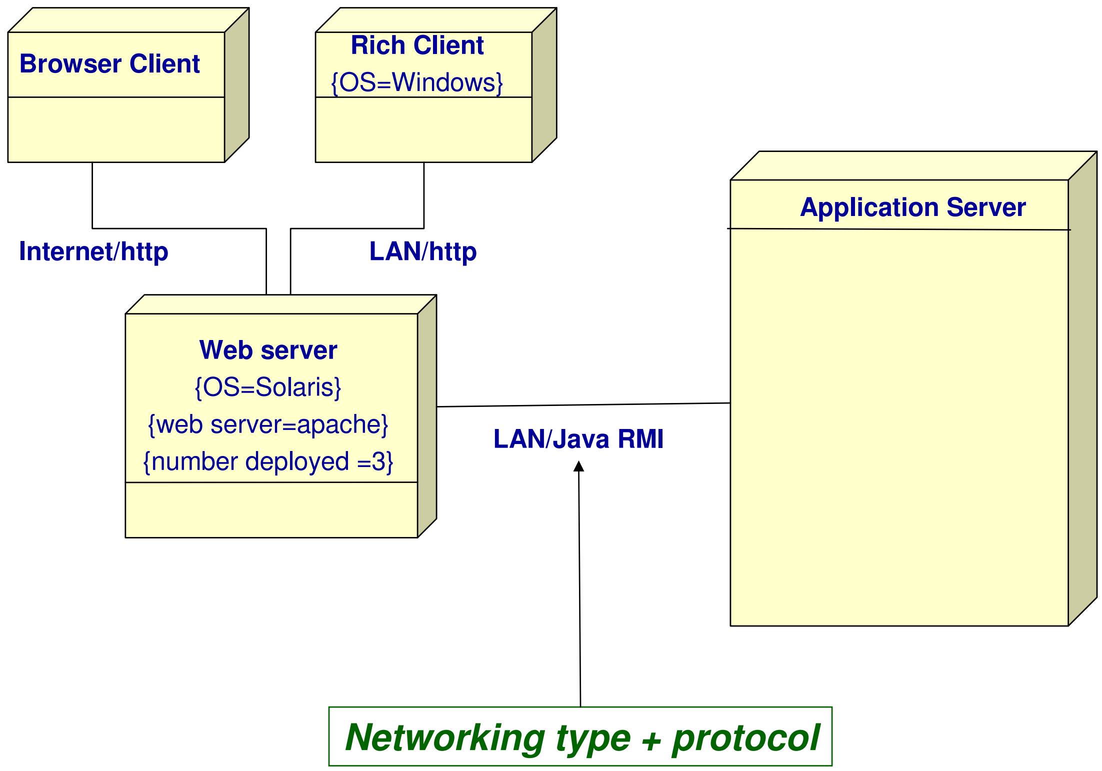
Deployment Diagrams> Connections



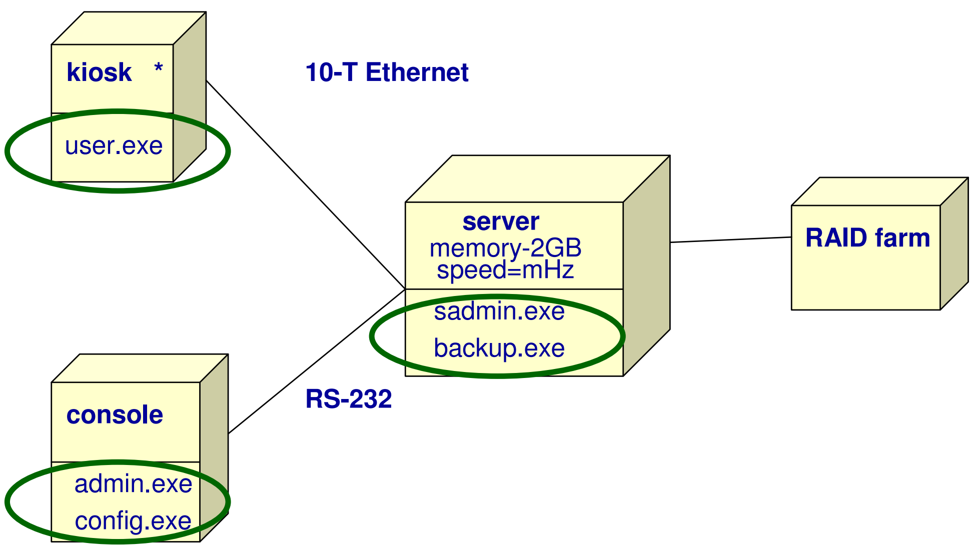
Connections

- Ethernet, serial line, satellite link
- we can use **stereotypes** to distinguish them to types
  - <<serial line>>
  - <<satellite link>>
  - …

Deployment Diagrams> Connections

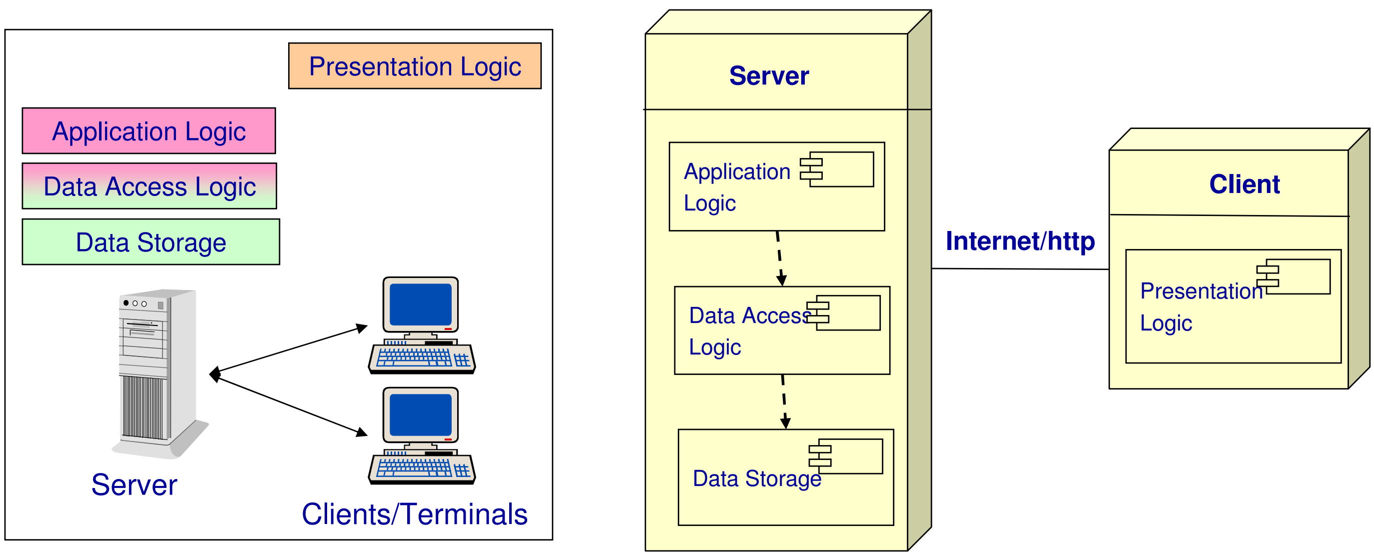


Modeling the Distribution of Artifacts



Combining Component and Deployment Diagrams

Combining Component and Deployment Diagrams: Example



Combining Component and Deployment Diagrams: Notes

- If we try to show all the components of a system in deployment diagrams they are will probably become very large and difficult to read.
- So we usually depict the key elements
- Alternatively, (in case we want to show everything) we can use a table to denote artifacts and their locations (e.g. use Excel)

Hardware and Software Specification

- We have to specify the new hardware or software that must be purchased
- Actual acquisition of hardware and software usually left to a purchasing department -- especially in larger firms

Realities in Infrastructure Design

- Most often the infrastructure will be already in place
- Coordination of infrastructure components is very complex
  - The application developer will need to coordinate with infrastructure specialists

Steps in Hardware and Software Specification

- Note hardware in low-level network model to create list of needed hardware
- Describe equipment in as much detail as possible
- Consider whether increased processing and traffic will absorb unused hardware capacity
- Note all software running on each hardware component

Hardware

- Commercial/Business
  - Mainframes, Commerial Minicomputers, Microcomputers (Wintel: Windows on Intel), Embedded Systems
- Technical/Engineering
  - Supercomputers, Workstations and Servers (Sun SPARC), Microcomputers, Embedded Systems

Some distinctions:

- Open vs Proprietary
  - Proprietary: available by only one vendor (higher prices, low interoperability)
  - Open: available from many vendors (better prices, better interoperability)
- Black-Box vs Glass-Box
  - Black- box: only the vendor has access to its internals (e.g. bank ATM)
  - Glass Box: internals are accessible by the user, may replaceable by other vendor
    - Free UNIX derivatives (Linux, BSD) on Intel x86 with source code are glass-box systems

Networking

- Local Area Network
  - short-distance (one building)
- Backbone
  - medium distance (campus)
- Wide Area Network
  - long-distance
- Remote Access
  - via phone / cable TV/satellite

Networking

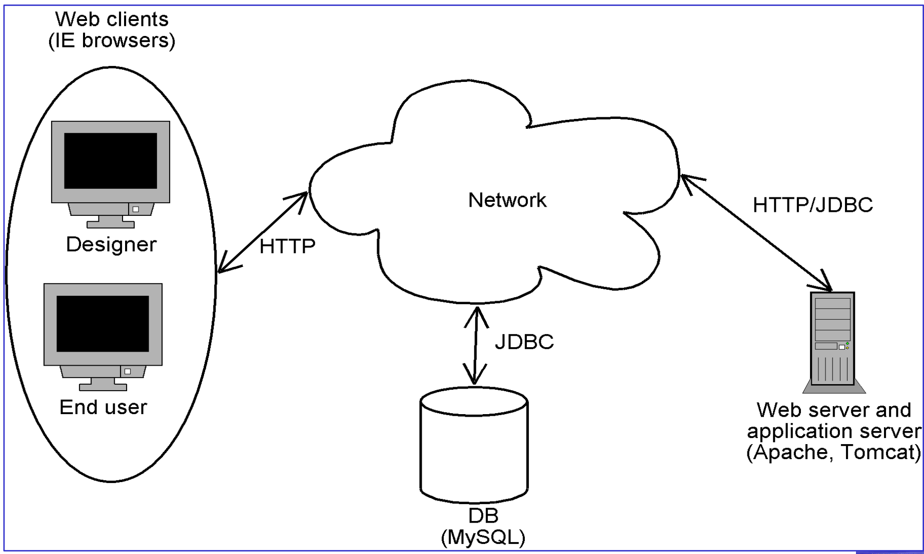
LAN	Backbone Network	WAN
• Ethernet – 10/100 Mb (1Gb fibre) – Inexpensive, widely used • Token Ring – 4/16 Mb – Not often used • ATM (copper) – 155 Mb (622Mb fibre) – Expensive, complex, flexible, high-overhead	• 100 Mb (fibre) or Gb Ethernet – fast, inexpensive, simple • FDDI – Old 100 Mbit (increasingly obsolete) • ATM – 155 Mb, 622 MB	• Long-distance line leased from telephone companies • Satellite links sometimes used

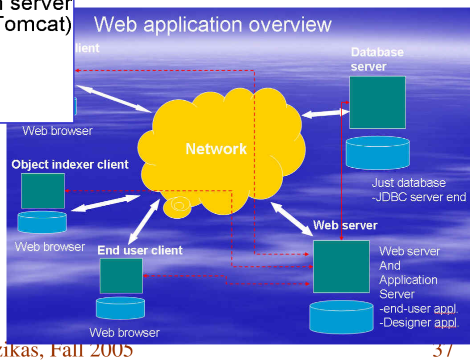
Remote Access

- Accessing a LAN or internet via phone/cable TV service
  - work from home, access when travelling, home internet service
  - Usually PPP over modem or cable modem
- DSL services

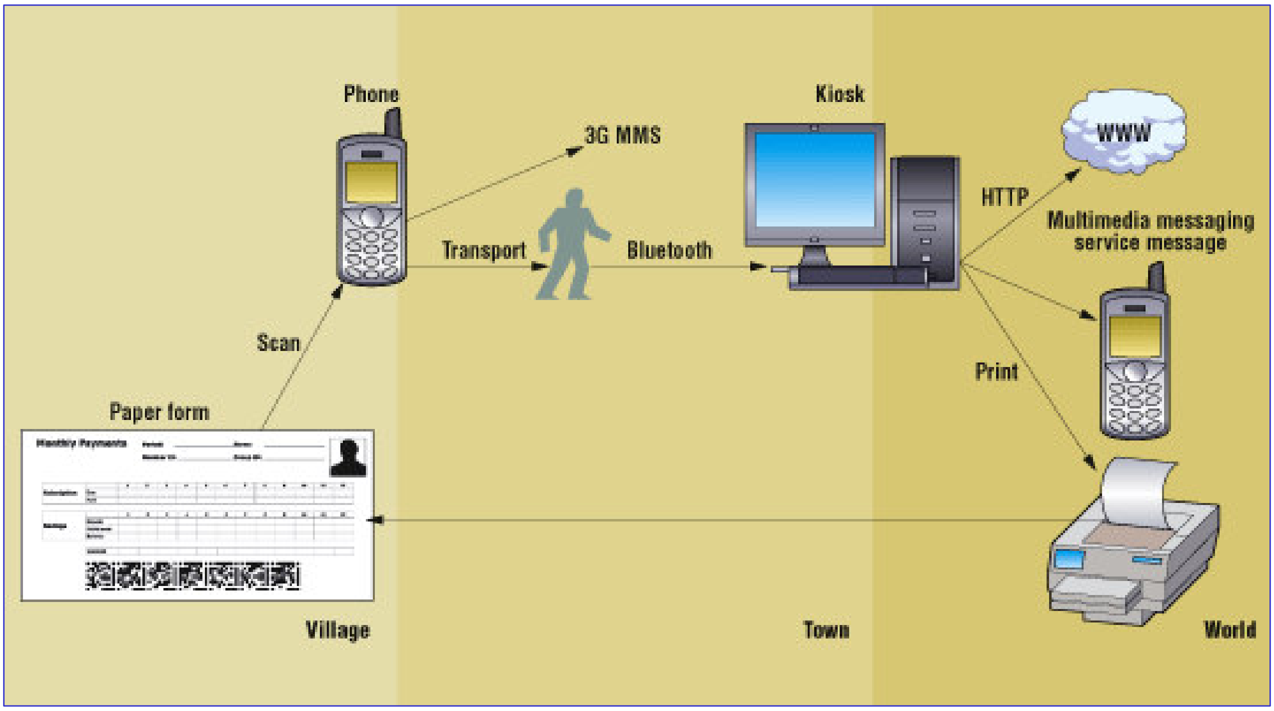
Deployment diagrams are usually depicted
in a <u>less formal and more liberal /vivid graphical notation</u>

Deployment Diagrams: Examples (Fastaxon)

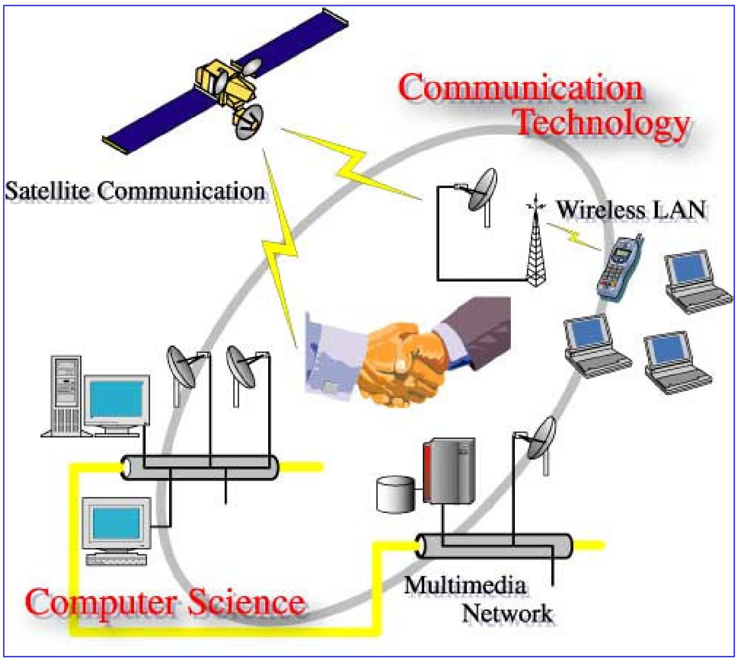




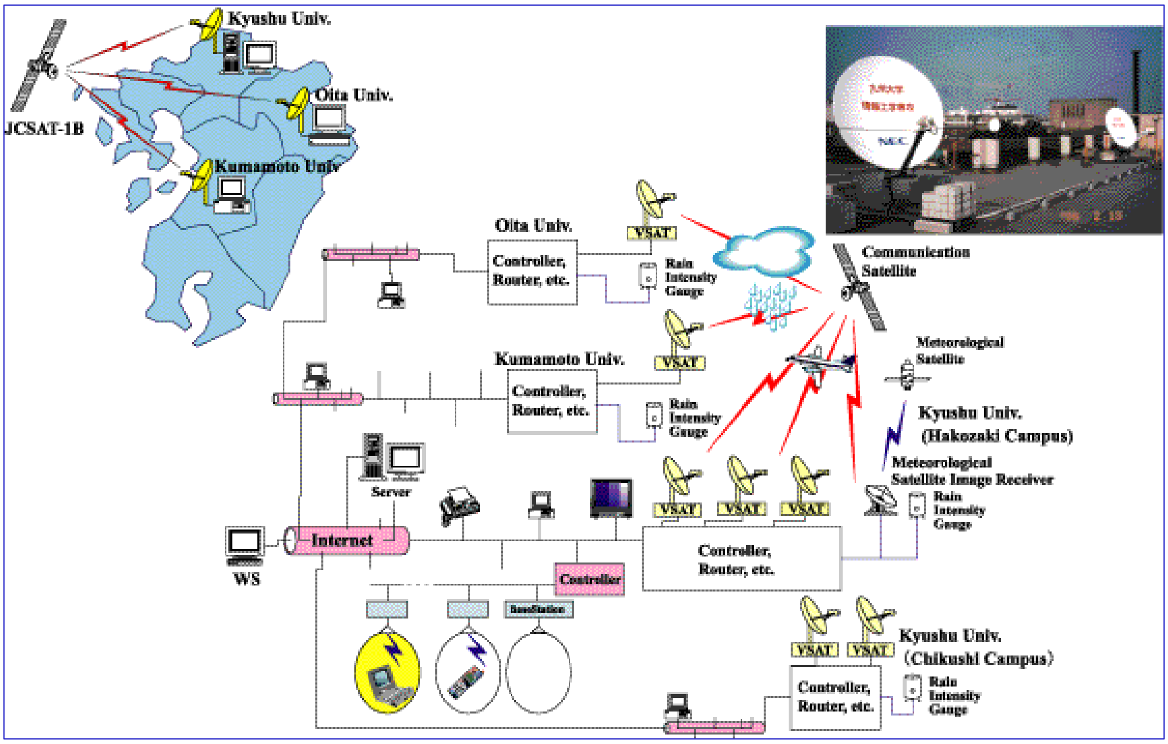
Deployment Diagrams: Examples



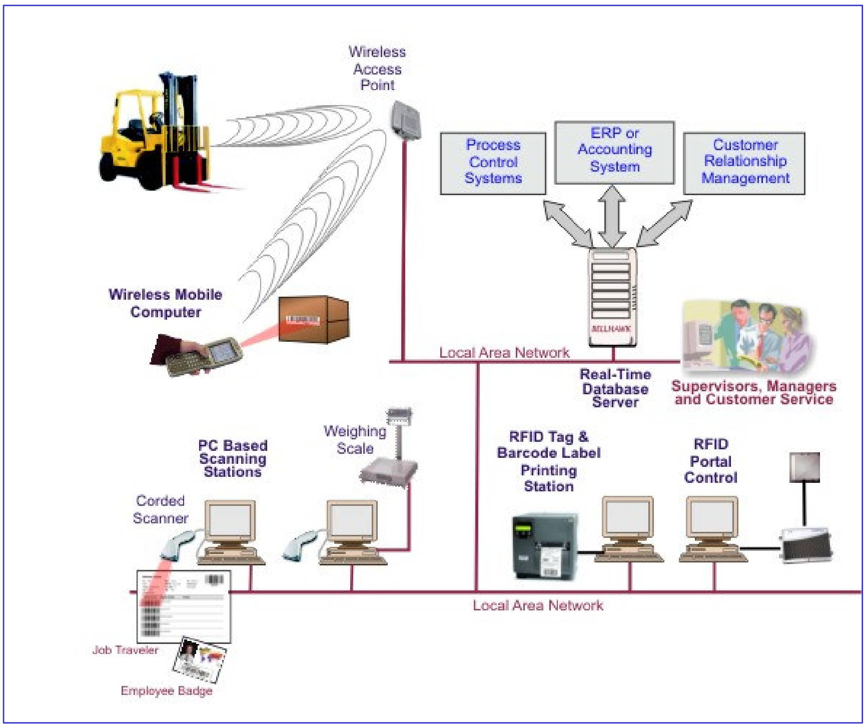
Deployment Diagrams: Examples



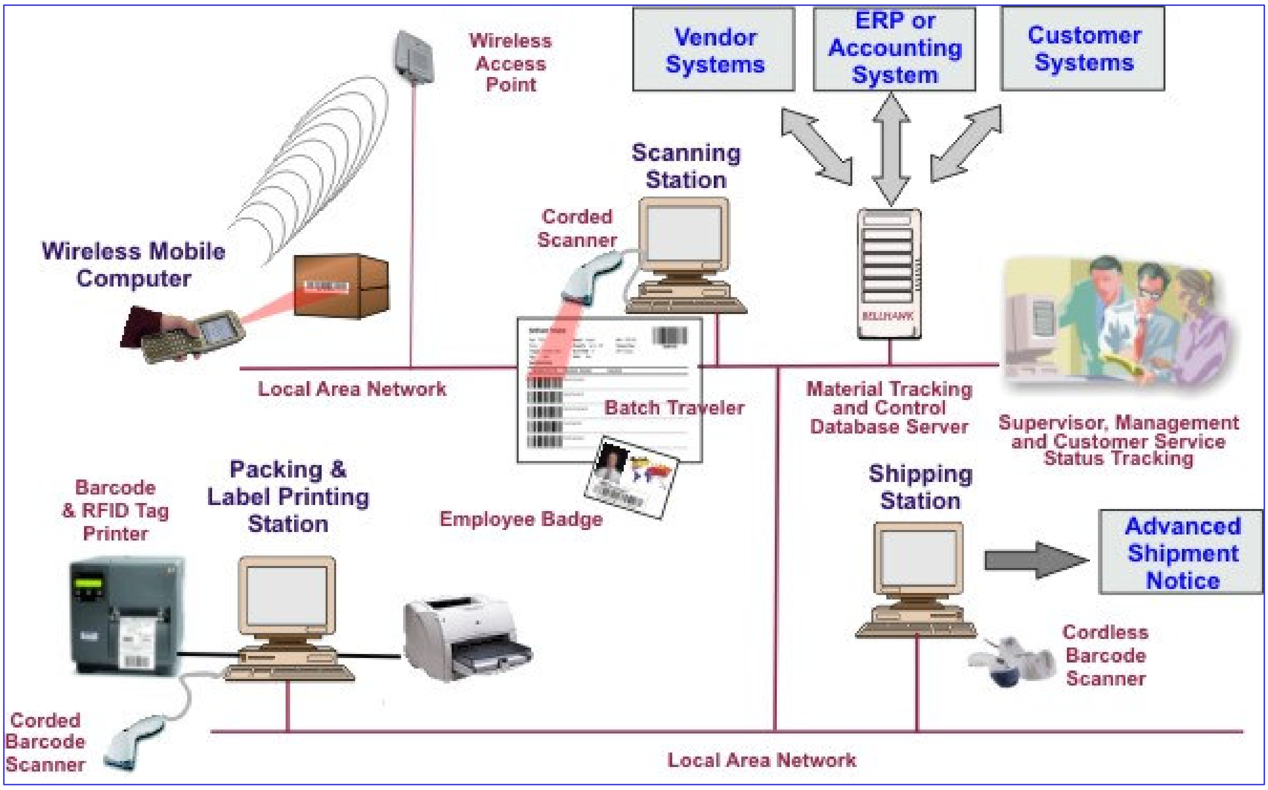
Deployment Diagrams: Examples



Deployment Diagrams: Examples



Deployment Diagrams: Examples



Deployment: Reading and References

- **UML Distilled: A Brief Guide to the Standard Object Modeling Language** (3rd Edition) by Martin Fowler, Addison Wesley, 2004. Chapter 8, Chapter 14 (2nd Edition: Chapter 10)
- **The Unified Modeling Language User Guide** (2nd edition) by G. Booch, J. Rumbaugh, I. Jacobson, Addison Wesley, 2004 **Chapter 27**
- **Requirements Analysis and System Design** (2nd edition) by Leszek A. Maciaszek, Addison Wesley, 2005, Chapter 6
- **Object-Oriented Systems Analysis and Design Using UML** (2nd edition) by S. Bennett, S. McRobb, R. Farmer, McGraw Hil, 2002 **, Chapter 19**
- http://www.agilemodeling.com/artifacts/componentDiagram.htm