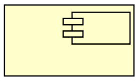
HY 351: Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
CS 351: Information Systems Analysis and Design

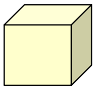
HY351:
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design

Physical (or Implementation) Diagrams

- UML **component** diagrams



- UML **deployment** diagrams



Γιάννης Τζίτζικας

Διάλεξη :
Ημερομηνία : 2008
Θέμα :

Διάρθρωση

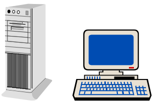
- Component Diagrams (Διαγράμματα Εξαρτημάτων)
- Deployment Diagrams (Διαγράμματα Παράταξης)
- Συνδυάζοντας διαγράμματα Εξαρτημάτων και Παράταξης

Βασικές ερωτήσεις
Key questions

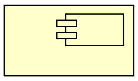
Οι υπολογιστικές πλατφόρμες αποτελούνται από υλικό, λογισμικό (PLs, DBMSs) και δικτύωση

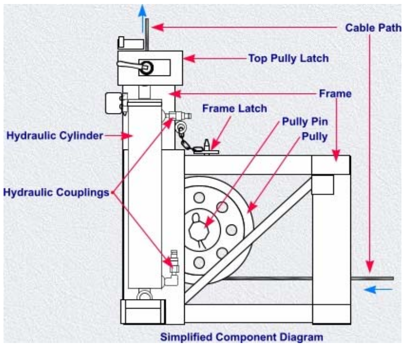
Ποια πλατφόρμα είναι πιο κατάλληλη για αυτό το πληροφοριακό σύστημα;

- *Πώς να επιλέξουμε το υλικό (hardware);*
- *Πώς να επιλέξουμε το λογισμικό (software);*
- *Πώς να επιλέξουμε τη δικτύωση (networking);*
- *Πώς να εκφράσουμε τη φυσική αρχιτεκτονική (μάθημα 18) του συστήματος με μια στάνταρτ διαγραμματική μορφή;*



Διαγράμματα **Εξαρτημάτων**
UML **Component** Diagrams





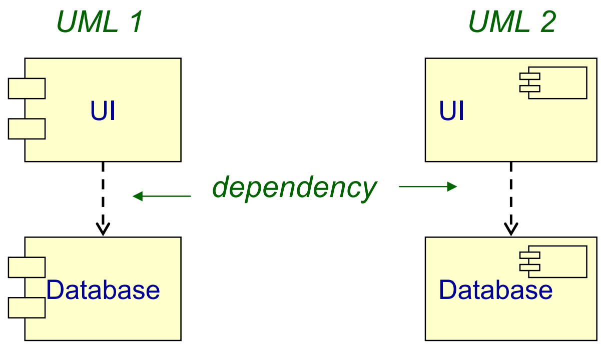
Simplified Component Diagram

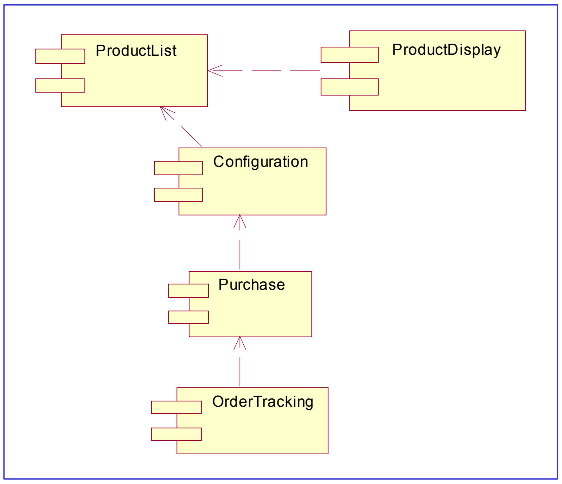
Component Diagrams (διαγράμματα εξαρτημάτων)

Component Diagrams show various components and their dependencies

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

Συμβολισμοί :





Τα χαρακτηριστικά ενός εξαρτήματος
The Characteristics of a Component

- a unit of independent deployment (never deployed partially)
- sufficiently documented and self-contained 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 replaceable part of a system (can be replaced by another component that conforms to the same interface)
- it fulfils a clear function and is logically and physically cohesive
- it may be nested in other components

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

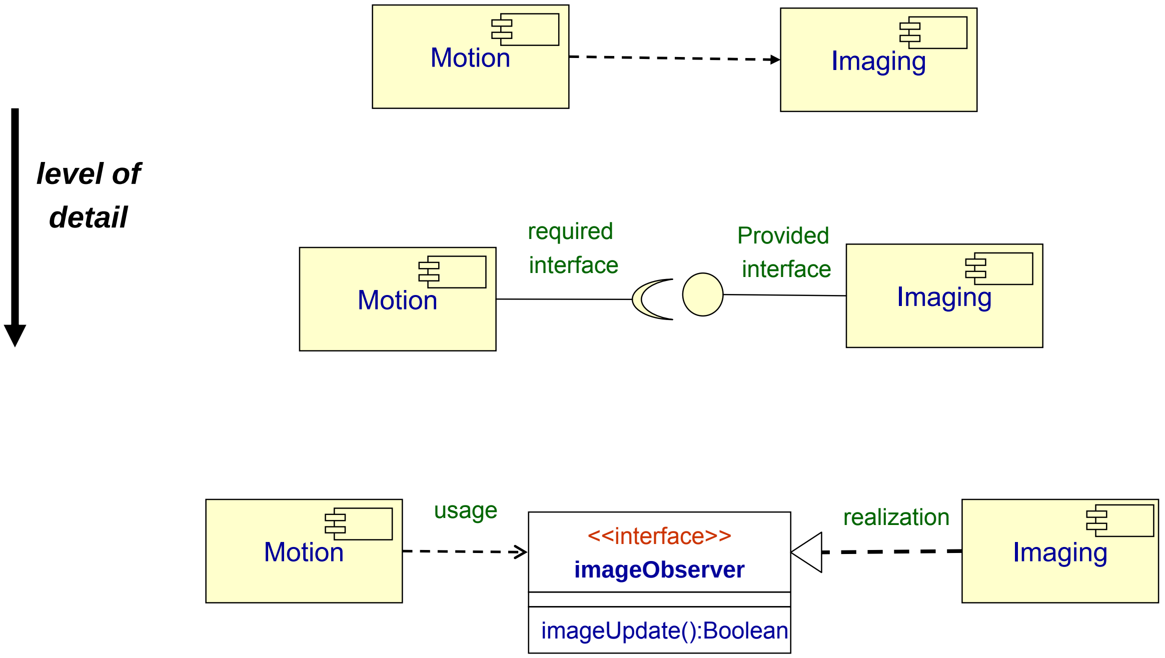
Εξαρτήματα
Components

- 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

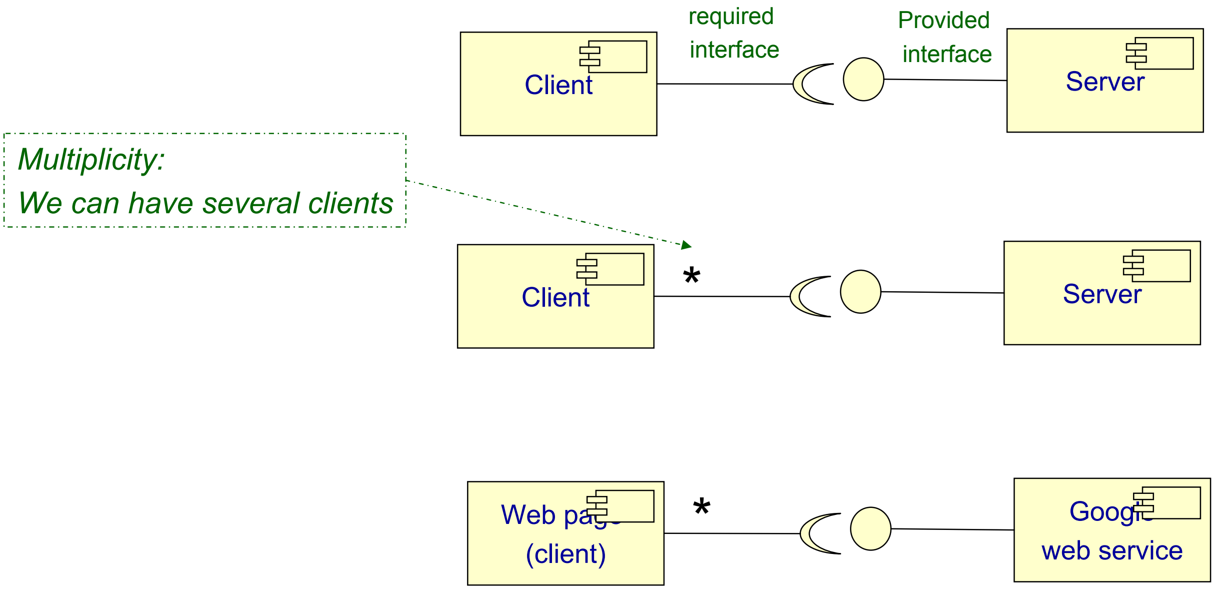
Εξαρτήματα και σχετικές έννοιες
Components and related Notions

- **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

Εξαρτήματα και Διεπαφές
Components and interfaces

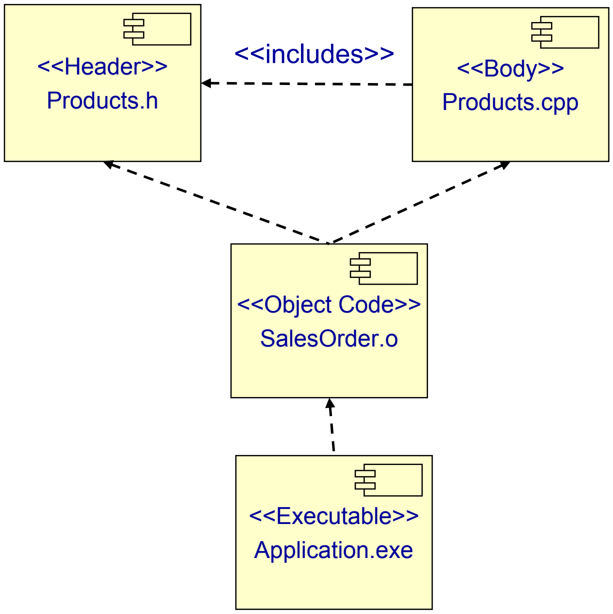


Εξαρτήματα και Διεπαφές (II)
Components and interfaces (II)

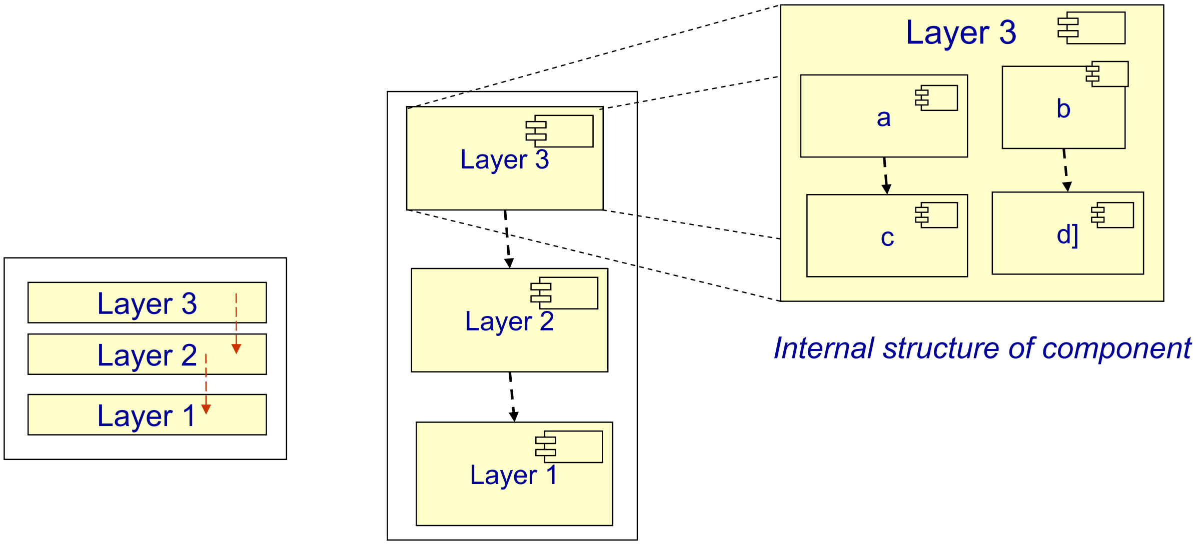


Παραδείγματα εξαρτημάτων
Fine-grained Components: Example

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

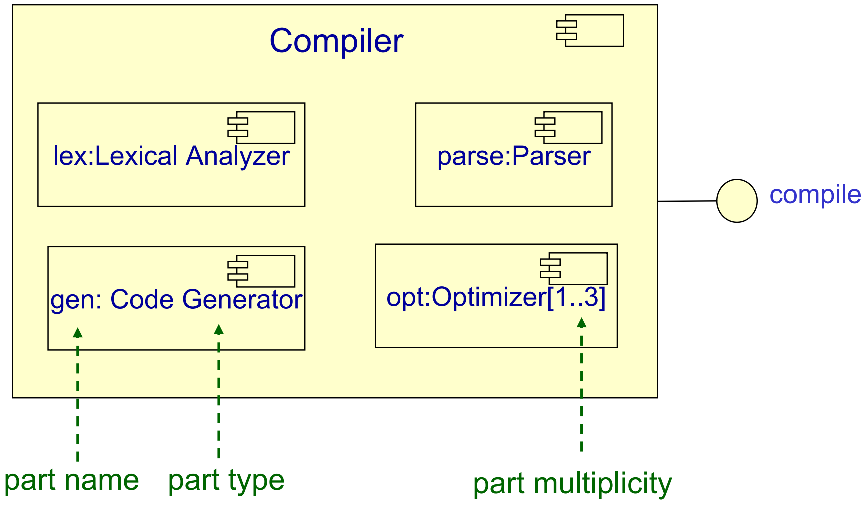


Coarse-grained components: e.g. Layers



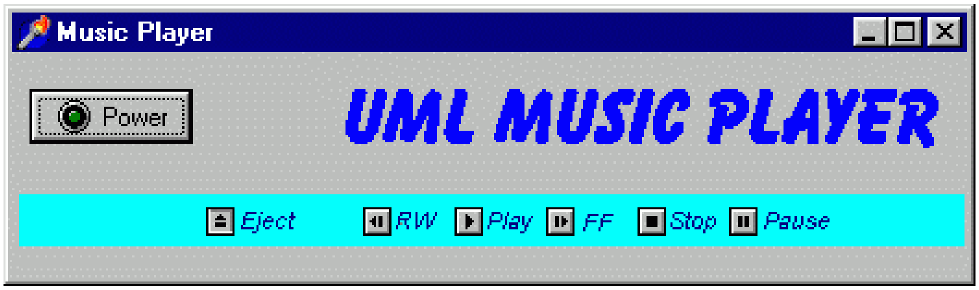
Internal structure of component

Εσωτερική Δομή Εξαρτημάτων
Internal Structure of Components



Παράδειγμα
πηγή:http://odl-skopje.etf.ukim.edu.mk/uml-help/

Suppose that we need to build up a software for playing a music from a CD-ROM Drive. A visual programming language might be used (VisualBasic or Delphi for example). If language supports multimedia controls, than we can use its components and reprogram them if necessary, or we can program new components. One possible graphical design for our player might be:



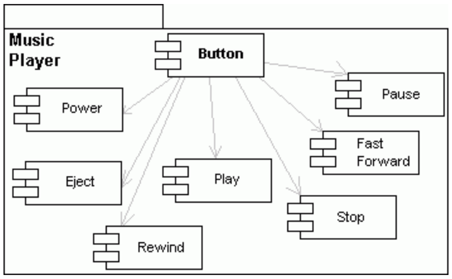
(συνέχεια)

As you can see this UML Music Player needs these controls:

- **play stop eject pause fast forward rewind power**

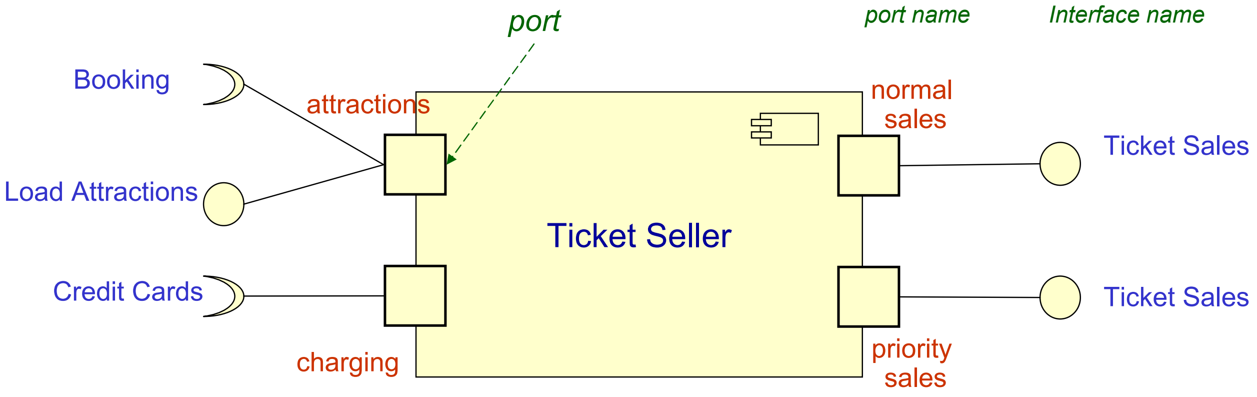
These controls will be realized by ***buttons***, thus we'll have a button performing these controls. If we look at buttons as separete components, we can draw out a component UML diagram. This is shown on the following picture.

All the components shown on the diagram belong to one global component - ***Button***, but actions they perform are diferent. We must obtain these actions by programming them.



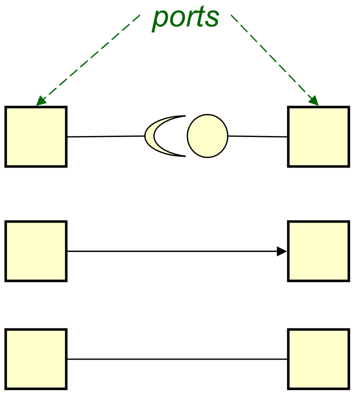
Ports

- 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.



Συνδέοντας Εξαρτήματα
Connecting Components

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



connector by interfaces

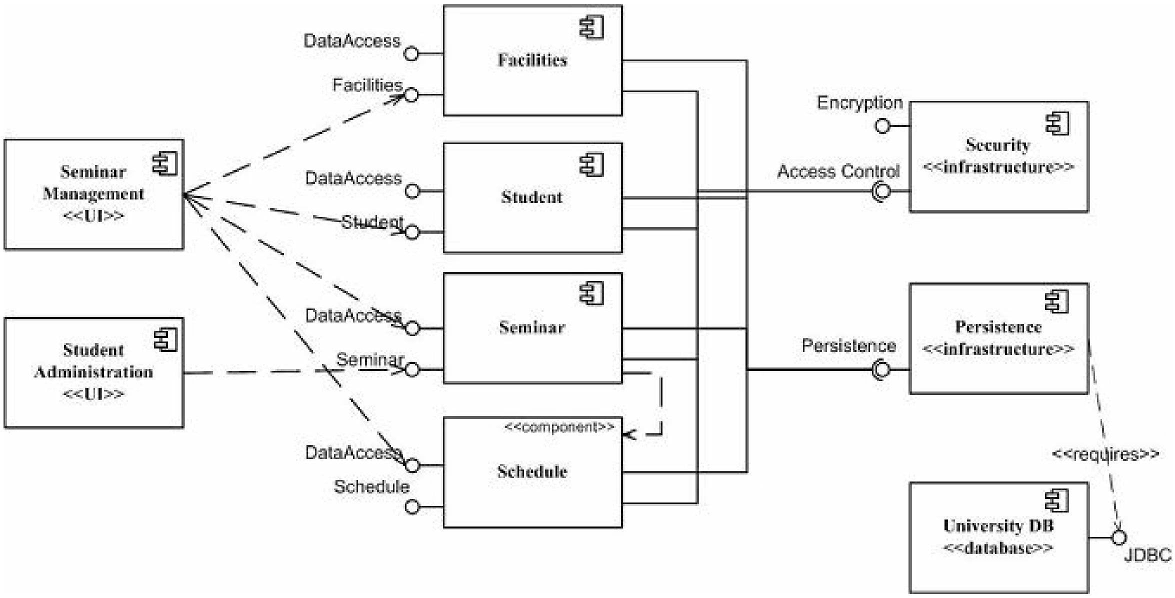
delegation connector

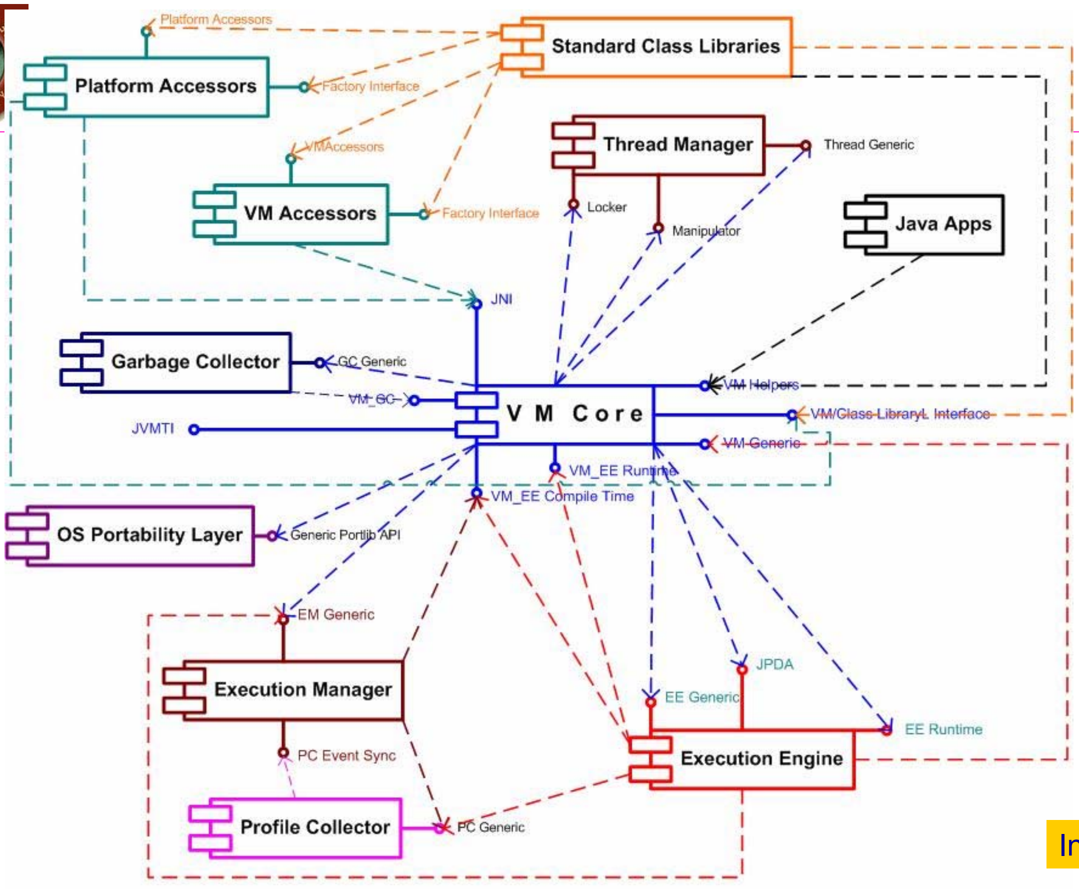
(connect an external port with the port of a part component)

direct connector

(more tight coupling)

some more examples





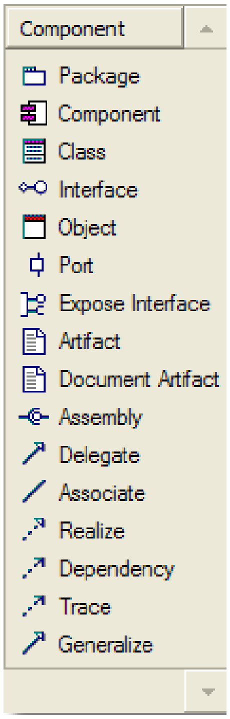
In UML 1

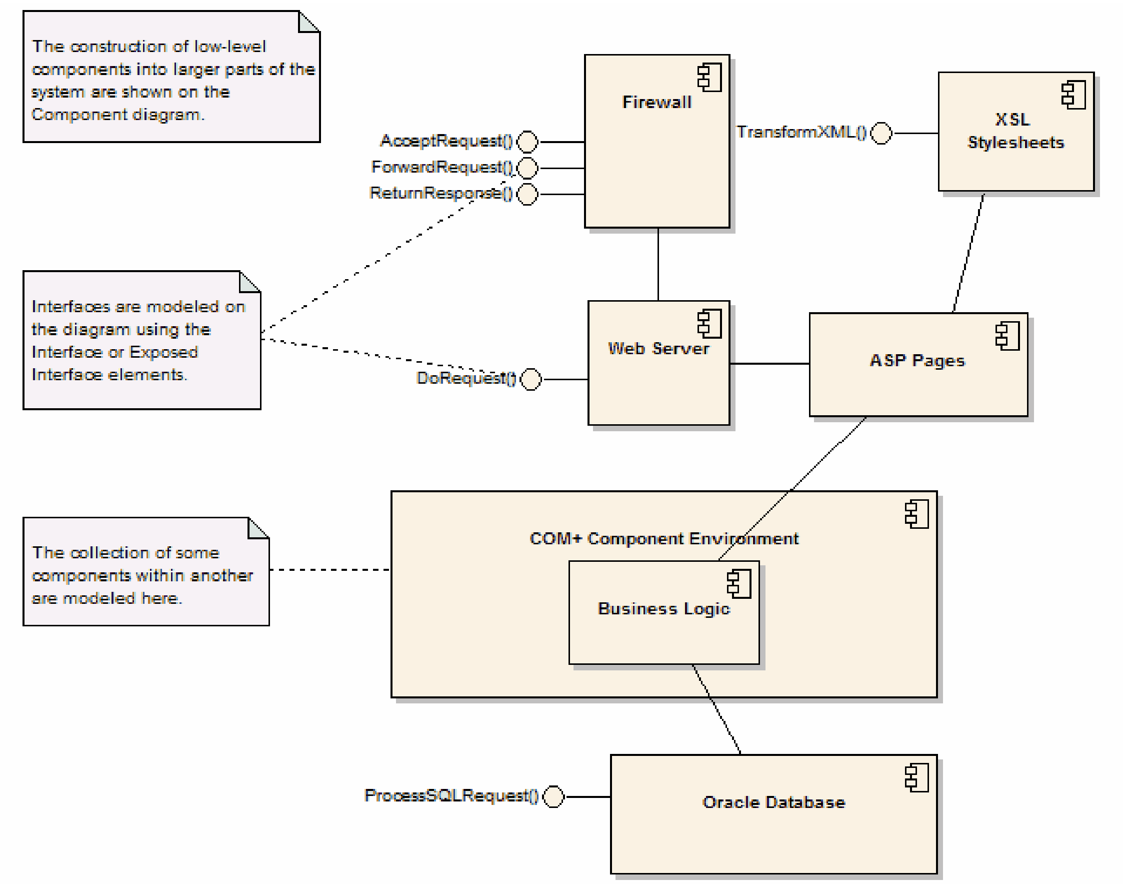
Modular Virtual Machine Interfaces Component Diagram

Case Study: Component Diagrams in Enterprise Architect

Component Diagrams in EA

- The **Component Model** defines how classes, artifacts and other low level elements are collected into high level components, and the interfaces and connections between them.
- **Components** are compiled software artifacts that work together to provide the required behaviour within the operating constraints defined in the requirements model.

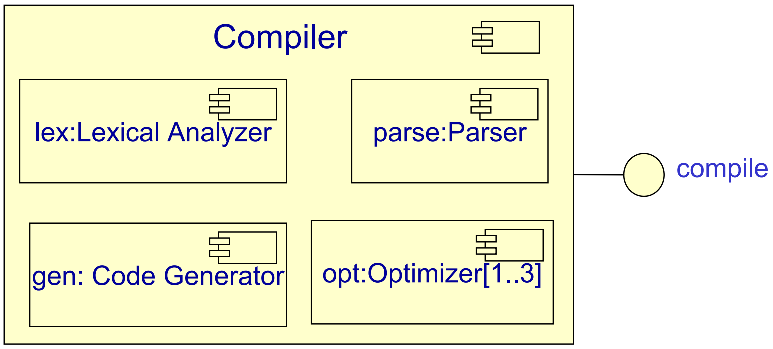


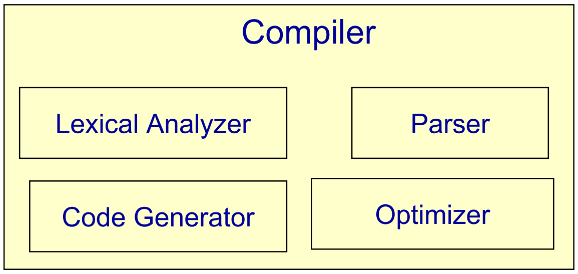

The construction of low-level components into larger parts of the system are shown on the Component diagram.

Interfaces are modeled on the diagram using the Interface or Exposed Interface elements.

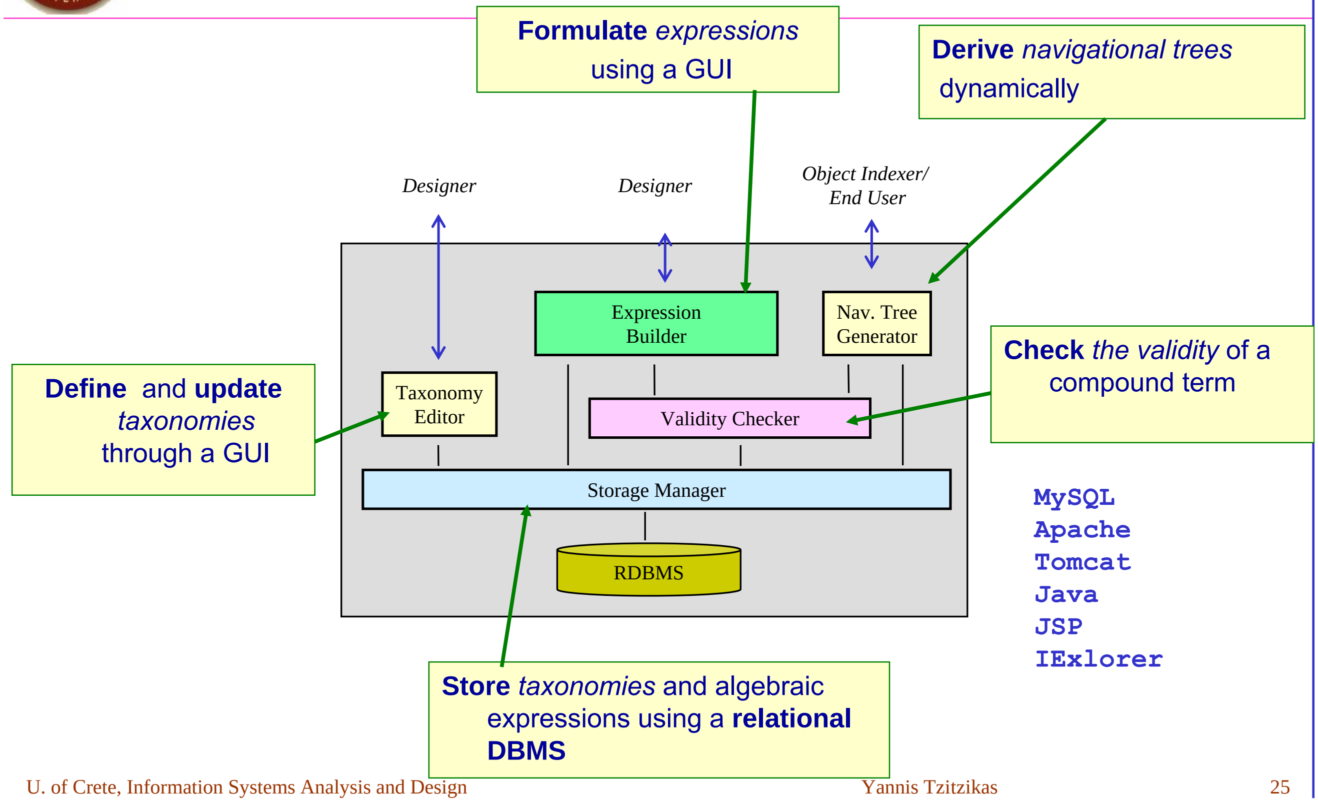
The collection of some components within another are modeled here.

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





FASTAXON (functional) architecture



Formulate *expressions* using a GUI

Derive *navigational trees* dynamically

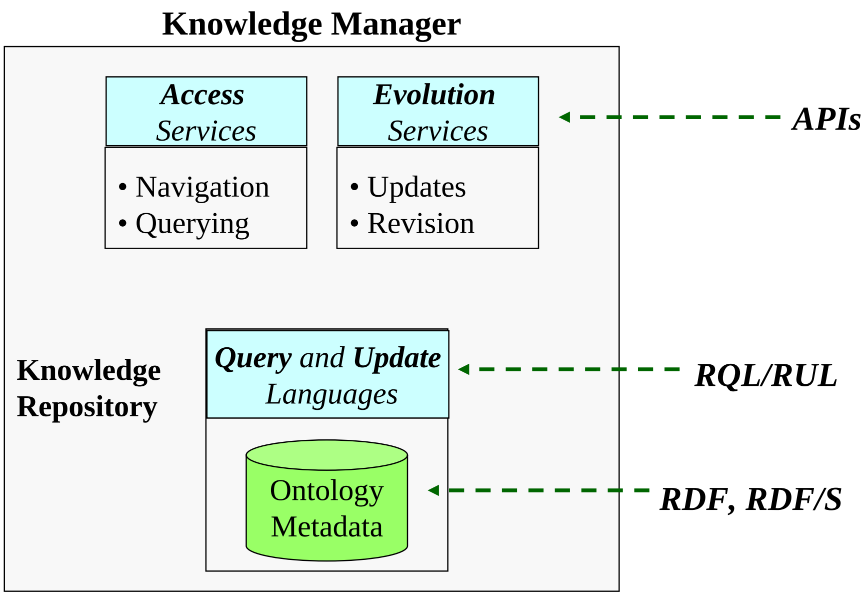
Define and **update** *taxonomies* through a GUI

Check *the validity* of a compound term

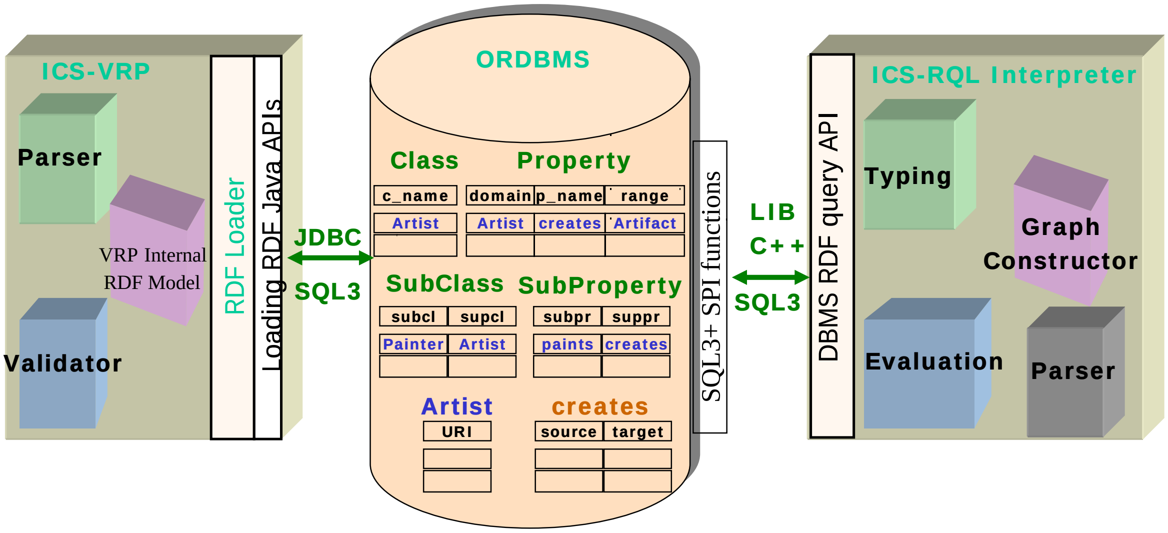
Store *taxonomies* and algebraic expressions using a **relational DBMS**

MySQL
Apache
Tomcat
Java
JSP
IExlorer

Knowledge Manager



RDF Suite Architecture



ICS-VRP

Parser

VRP Internal RDF Model

Validator

RDF Loader

Loading RDF Java APIs

JDBC

SQL3

ORDBMS

Class
c_name
Artist

Property		
domain	p_name	range
Artist	creates	Artifact

SubClass	
subcl	supcl
Painter	Artist

SubProperty	
subpr	suppr
paints	creates

Artist
URI

creates	
source	target

SQL3+ SPI functions

LIB

C++

SQL3

DBMS RDF query API

ICS-RQL Interpreter

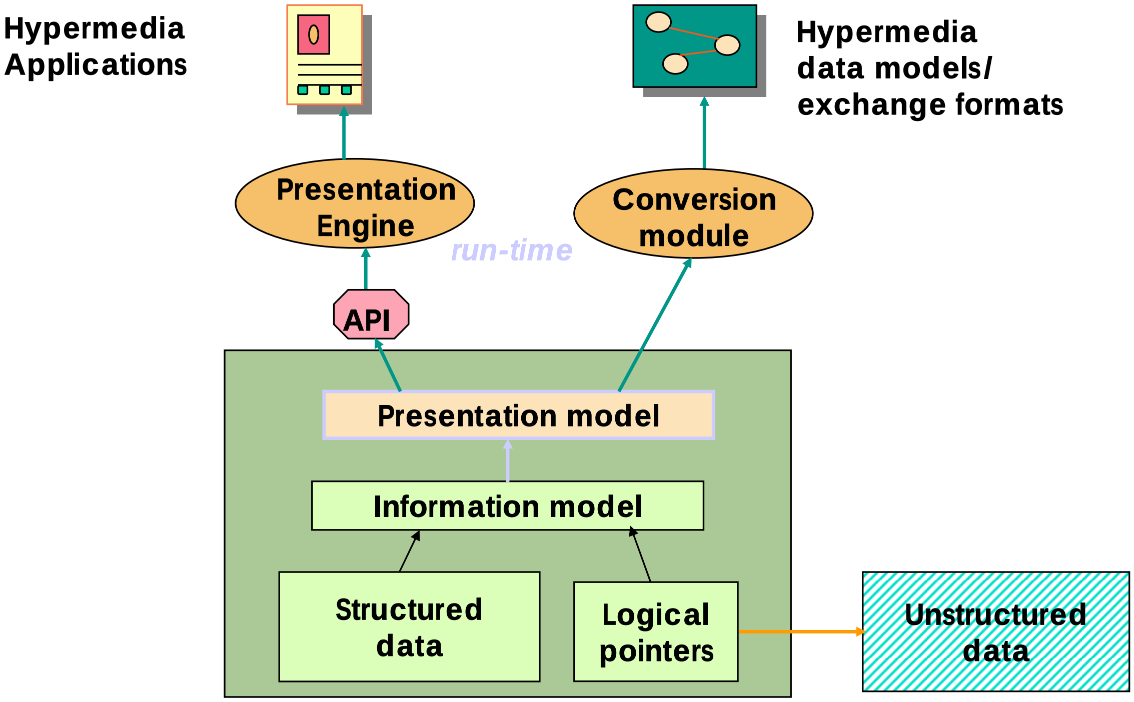
Typing

Graph Constructor

Evaluation

Parser

DOMENICUS Architecture



Hypermedia Applications

Hypermedia data models/ exchange formats

Presentation Engine

Conversion module

run-time

API

Presentation model

Information model

Structured data

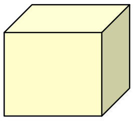
Logical pointers

Unstructured data

Semantic network-based Information Repository

OS/tool storage

Διαγράμματα Παράταξης
UML Deployment Diagrams

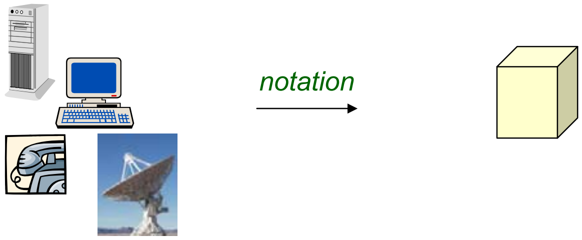


Deployment Diagrams
(διαγράμματα ανάπτυξης/σύνταξης/παράθεσης)

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

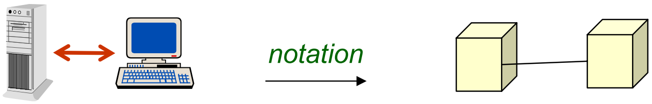
Node:

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

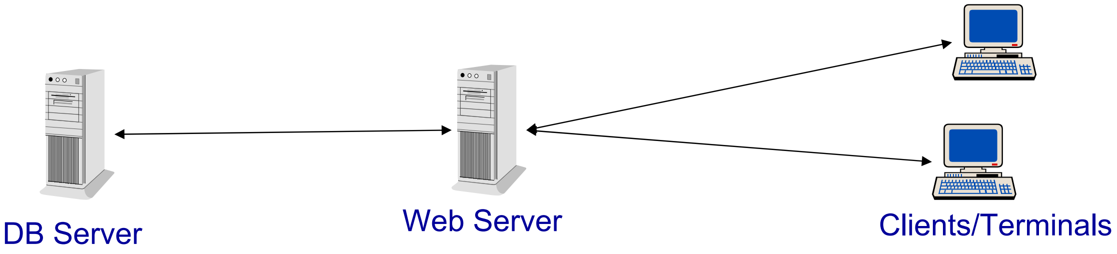


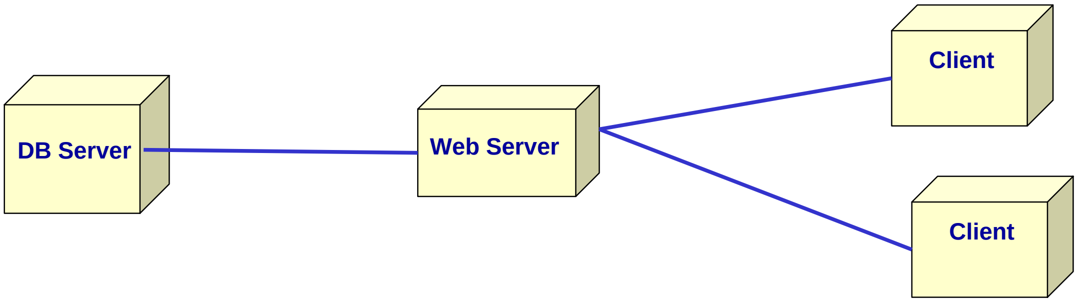
Connection (among nodes)

- communication paths over which the system will interact

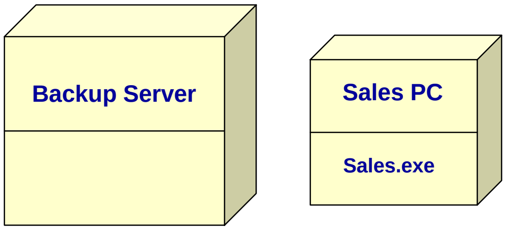


Ένα διάγραμμα παράταξης

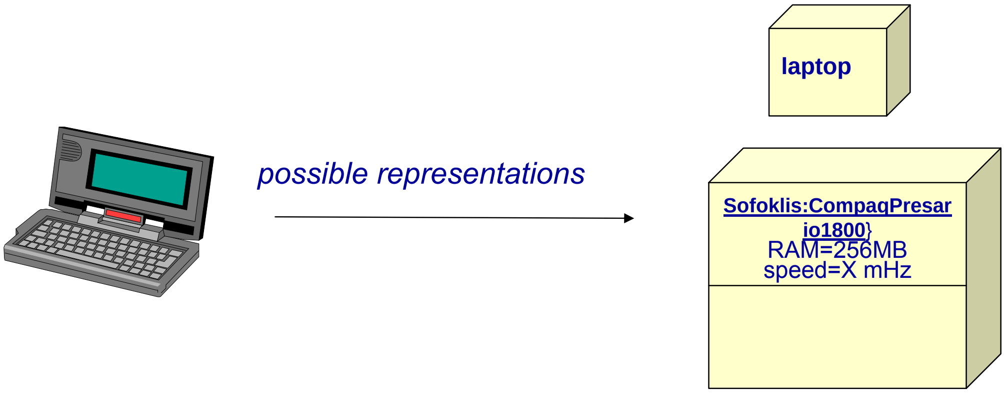




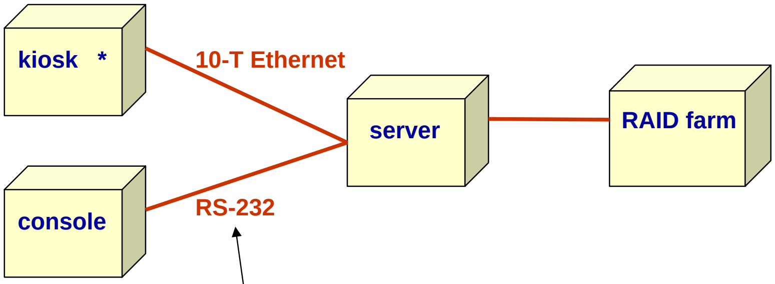
Deployment Diagrams> Nodes



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



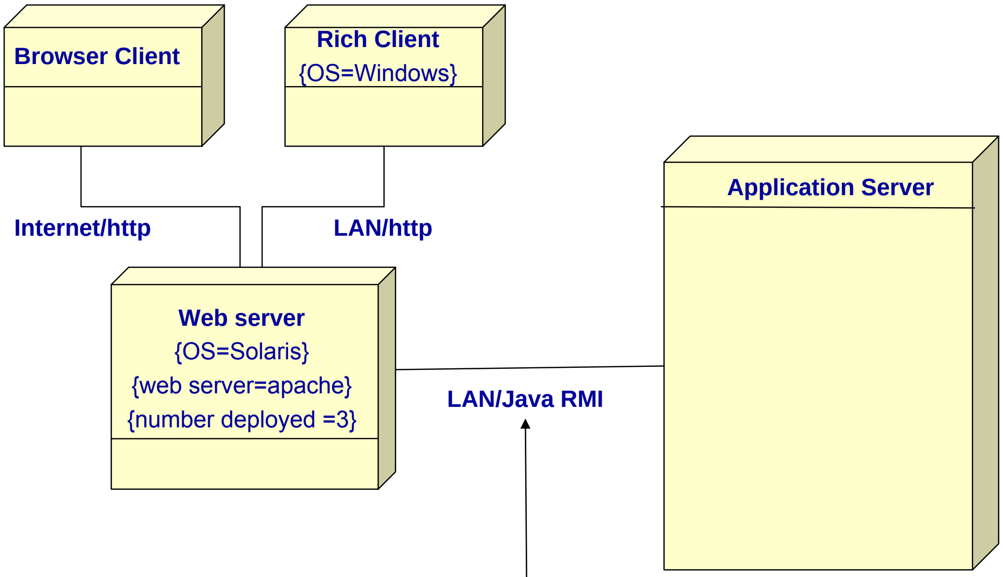
Deployment Diagrams> Connections



Connections

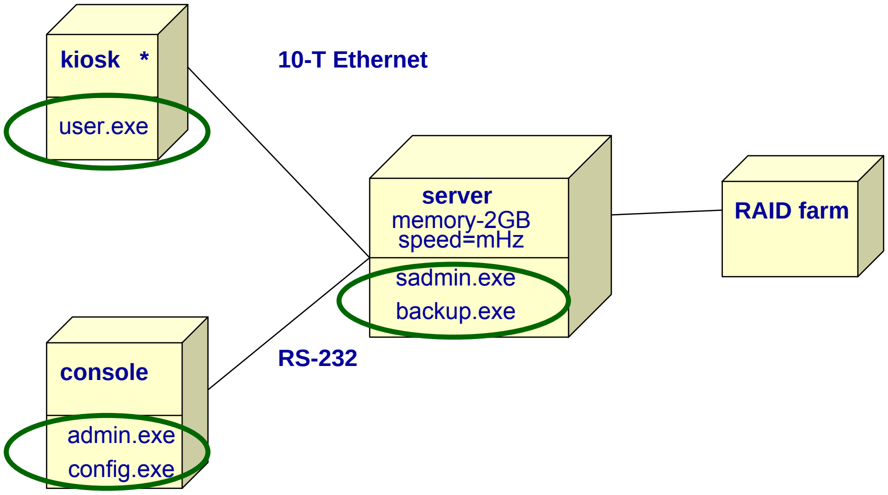
- Ethernet, serial line, satellite link
- we can use **stereotypes** to distinguish them to types
  - <<serial line>>
  - <<satellite link>>
  - ...

Deployment Diagrams> Connections



Networking type + protocol

Παριστάνοντας την κατανομή των τεχνουργημάτων
Modeling the Distribution of Artifacts



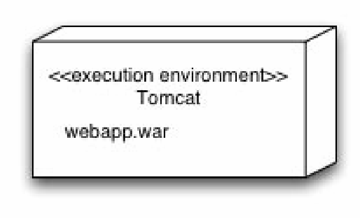
Example

- A WAR file (short for Web ARchive) could be a JAR file used to distribute a collection of JavaServer Pages, servlets, Java classes, XML files, tag libraries and static Web pages (HTML and related files) that together constitute a Web application.
  - A WAR file may be digitally signed in the same way as a JAR file in order to assert that the code is trusted.

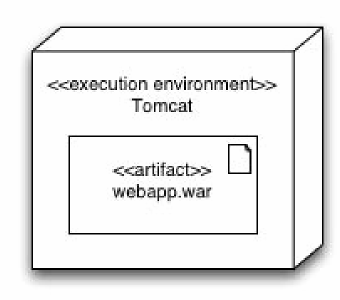
- There are special files and directories within a WAR file.
  - The /WEB-INF directory in the WAR file contains a file named web.xml which defines the structure of the web application. If the web application is only serving JSP files, the web.xml file is not strictly necessary. If the web application uses servlets, then the servlet container uses web.xml to ascertain which servlet a URL request should be routed to. web.xml is also used to define context variables which can be referenced within the servlets and it is used to define environmental dependencies which the deployer is expected to set up. An example of this is a dependency on a mail session used to send email. The servlet container is responsible for providing this service.

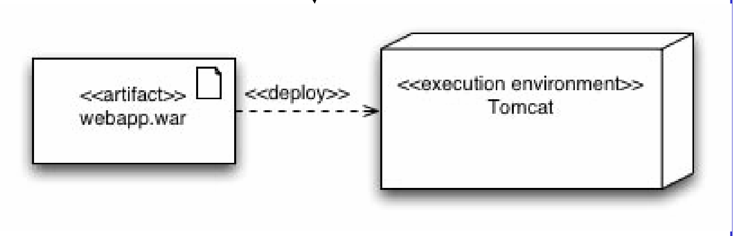
Example (cont)

Create a deployment diagram that shows a Java web application where the web application web archive jar is deployed to a Tomcat application server.



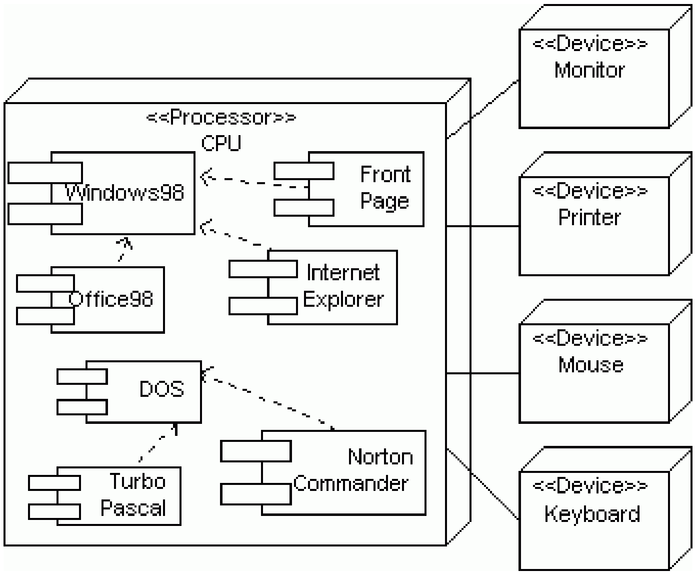
Any of the following 3 drawings is correct

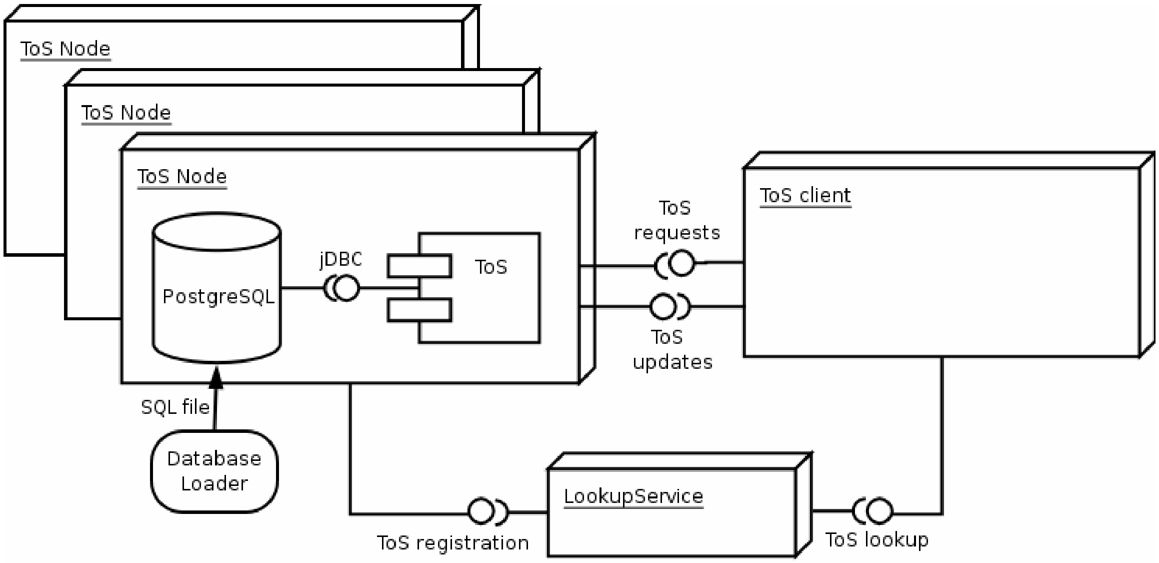




Συνδυάζοντας διαγράμματα Εξαρτημάτων και Παράταξης
Combining Component and Deployment Diagrams

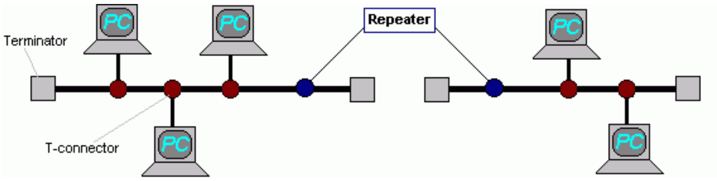
Παράδειγμα
πηγή http://odl-skopje.etf.ukim.edu.mk/uml-help/

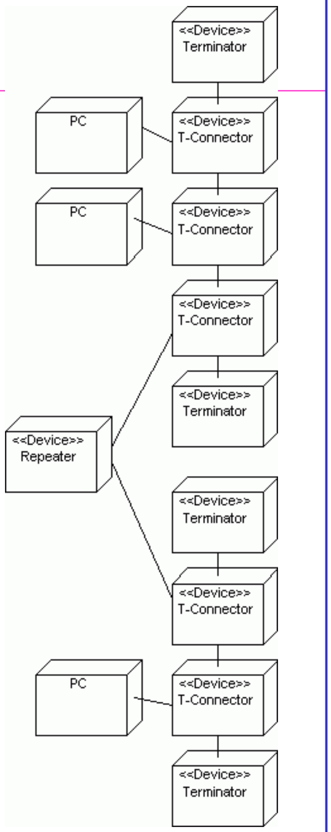




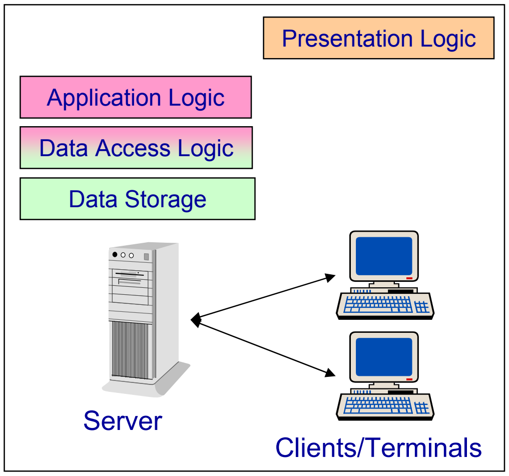
Another example:
http://odl-skopje.etf.ukim.edu.mk/uml-help/

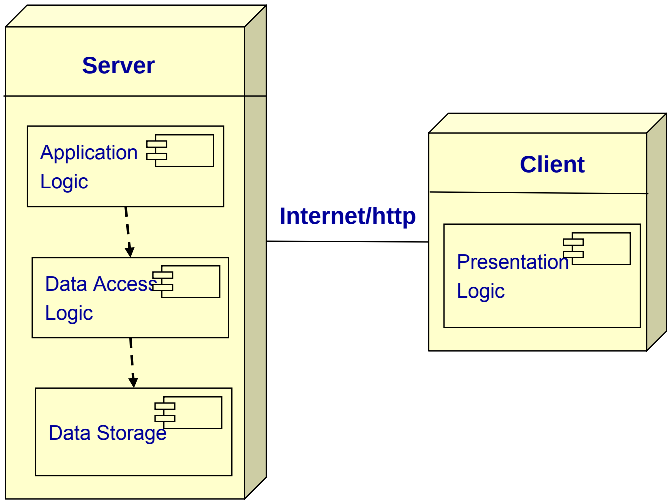
- Deployment diagram for ETHERNET

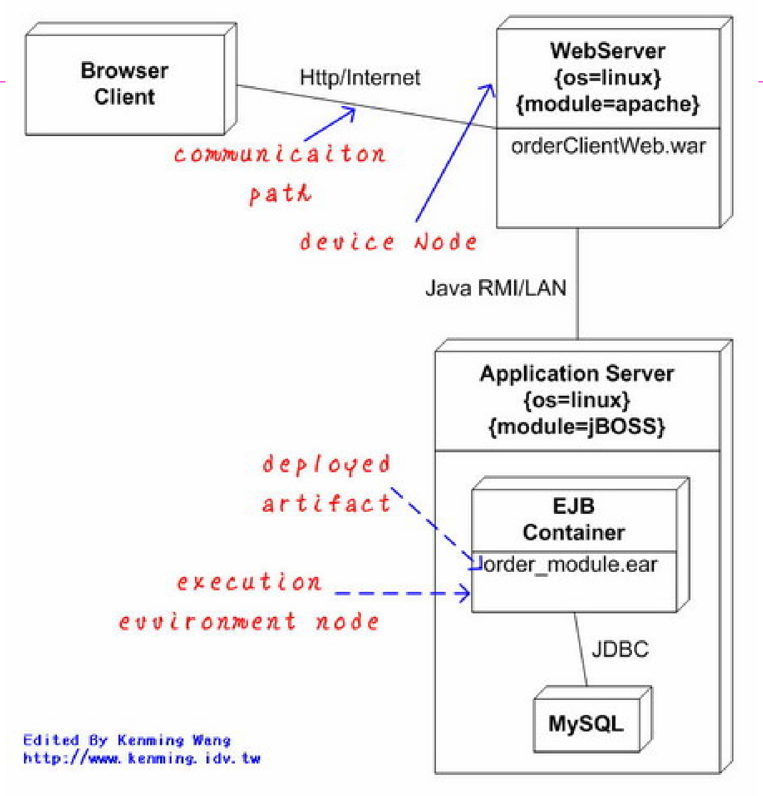




Combining Component and Deployment Diagrams:
Example







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

Δικτύωση
Networking

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

Δικτύωση
Networking

LAN

- 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

Backbone Network

- 100 Mb (fibre) or Gb Ethernet
  - fast, inexpensive, simple
- FDDI
  - Old 100 Mbit (increasingly obsolete)
- ATM
  - 155 Mb, 622 MB

WAN

- Long-distance line leased from telephone companies
- Satellite links sometimes used

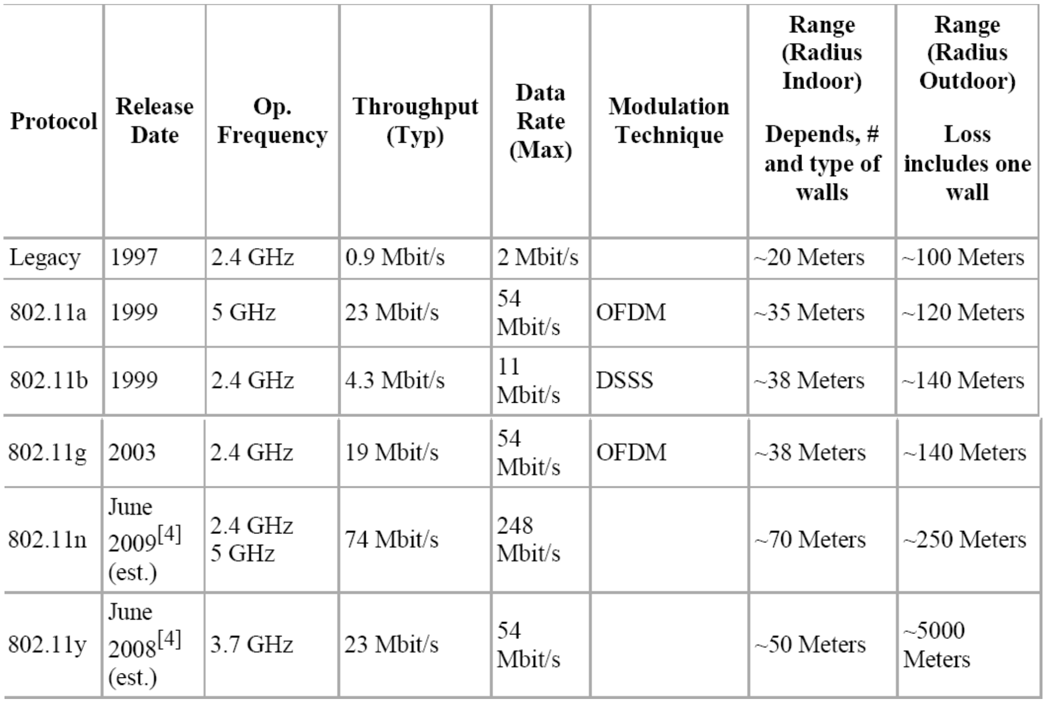
Remote Access

- 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

Wireless

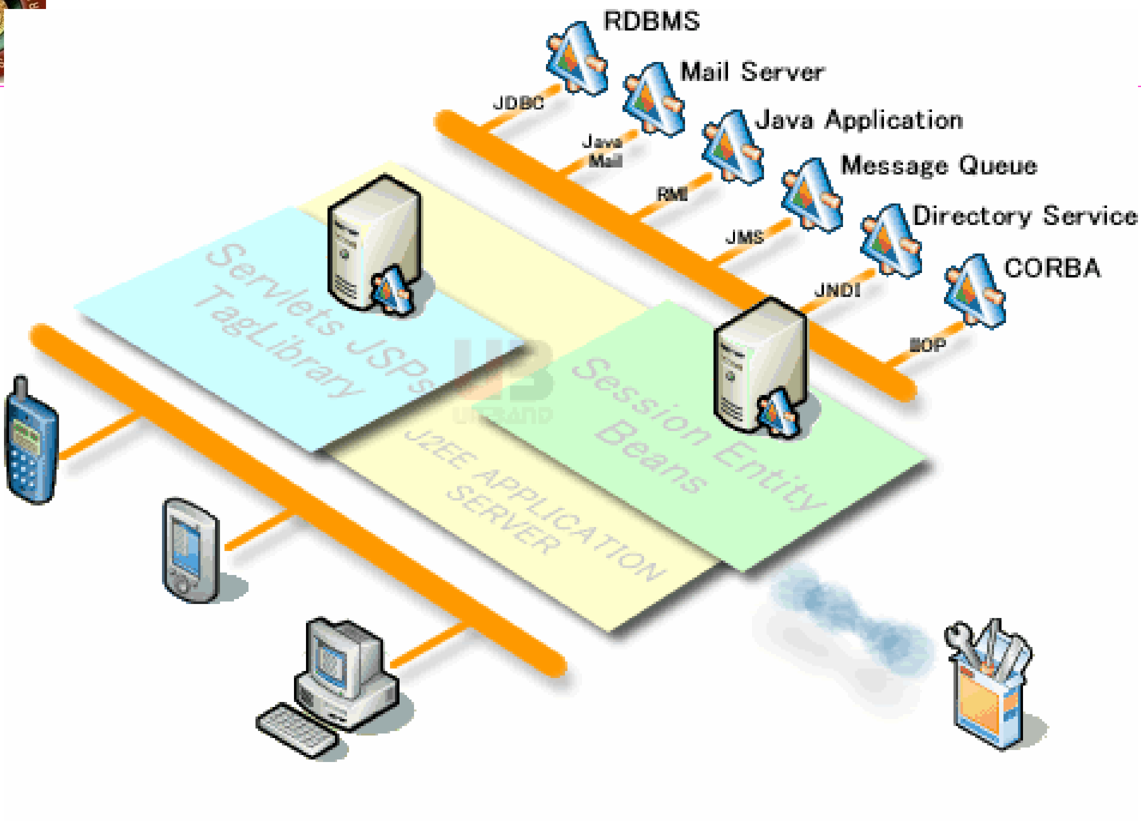
- IEEE 802.11 is a set of standards for wireless local area network (WLAN) computer communication in the 5 GHz and 2.4 GHz public spectrum bands.
- Although the terms 802.11 and Wi-Fi are often used interchangeably, the Wi-Fi Alliance uses the term "Wi-Fi" to define a slightly different set of overlapping standards.
- 802.11b and 802.11g use the 2.4 GHz ISM band, operating in the United States under Part 15 of the US Federal Communications Commission Rules and Regulations. Because of this choice of frequency band, 802.11b and g equipment may occasionally suffer interference from microwave ovens and cordless telephones. Bluetooth devices, while operating in the same band, in theory do not interfere with 802.11b/g because they use a frequency hopping spread spectrum signaling method (FHSS) while 802.11b/g uses a direct sequence spread spectrum signaling method (DSSS). 802.11a uses the 5 GHz U-NII band, which offers 8 non-overlapping channels rather than the 3 offered in the 2.4GHz ISM frequency band.

Wireless (cont)

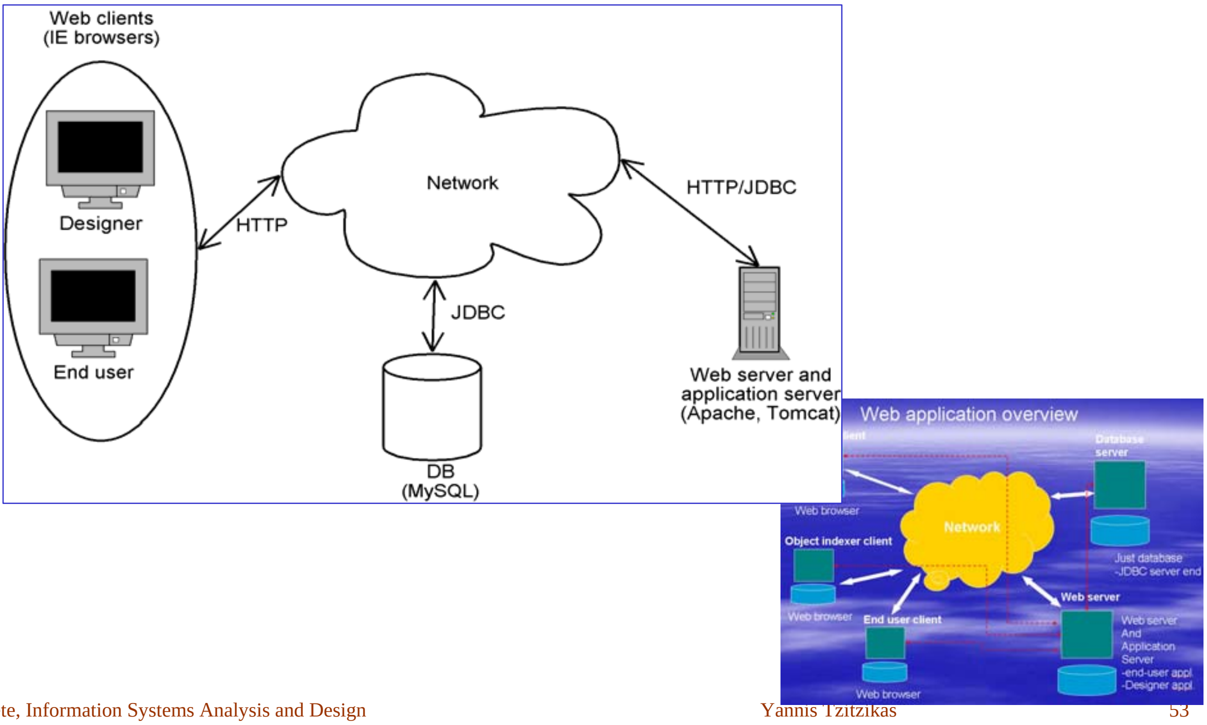


Protocol	Release Date	Op. Frequency	Throughput (Typ)	Data Rate (Max)	Modulation Technique	Range (Radius Indoor) Depends, # and type of walls	Range (Radius Outdoor) Loss includes one wall
Legacy	1997	2.4 GHz	0.9 Mbit/s	2 Mbit/s		~20 Meters	~100 Meters
802.11a	1999	5 GHz	23 Mbit/s	54 Mbit/s	OFDM	~35 Meters	~120 Meters
802.11b	1999	2.4 GHz	4.3 Mbit/s	11 Mbit/s	DSSS	~38 Meters	~140 Meters
802.11g	2003	2.4 GHz	19 Mbit/s	54 Mbit/s	OFDM	~38 Meters	~140 Meters
802.11n	June 2009[4] (est.)	2.4 GHz 5 GHz	74 Mbit/s	248 Mbit/s		~70 Meters	~250 Meters
802.11y	June 2008[4] (est.)	3.7 GHz	23 Mbit/s	54 Mbit/s		~50 Meters	~5000 Meters

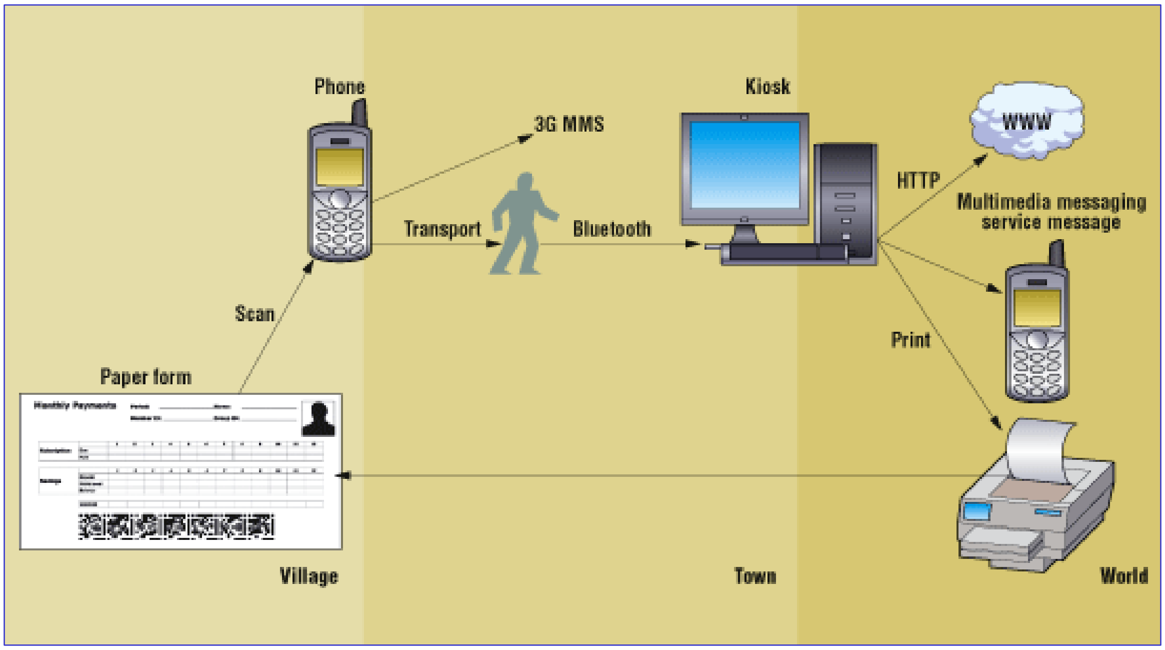
Deployment diagrams are usually depicted
in a <u>less formal and more liberal /vivid graphical notation</u>



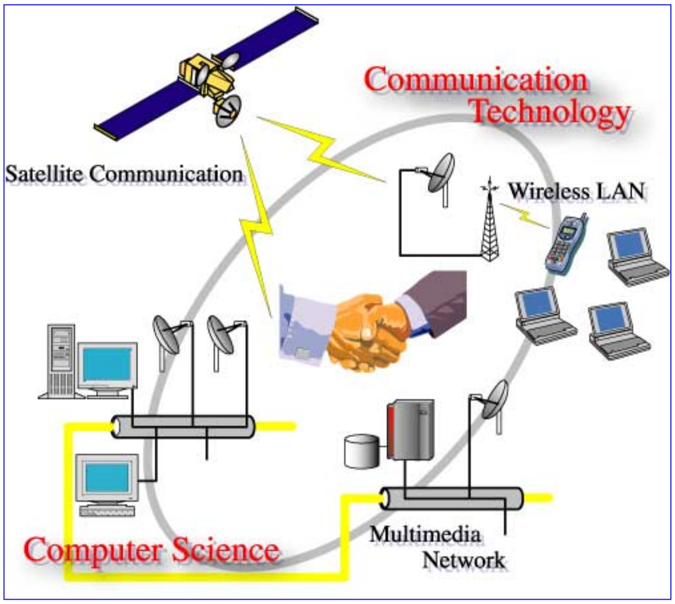
Deployment Diagrams: Examples (Fastaxon)



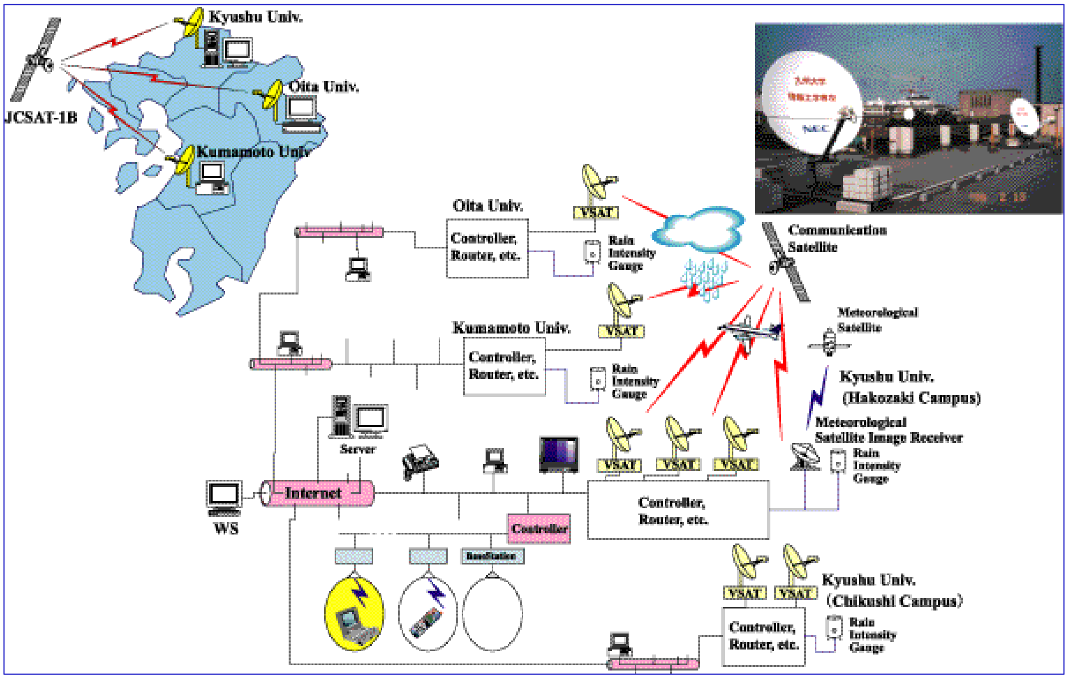
Deployment Diagrams: Examples



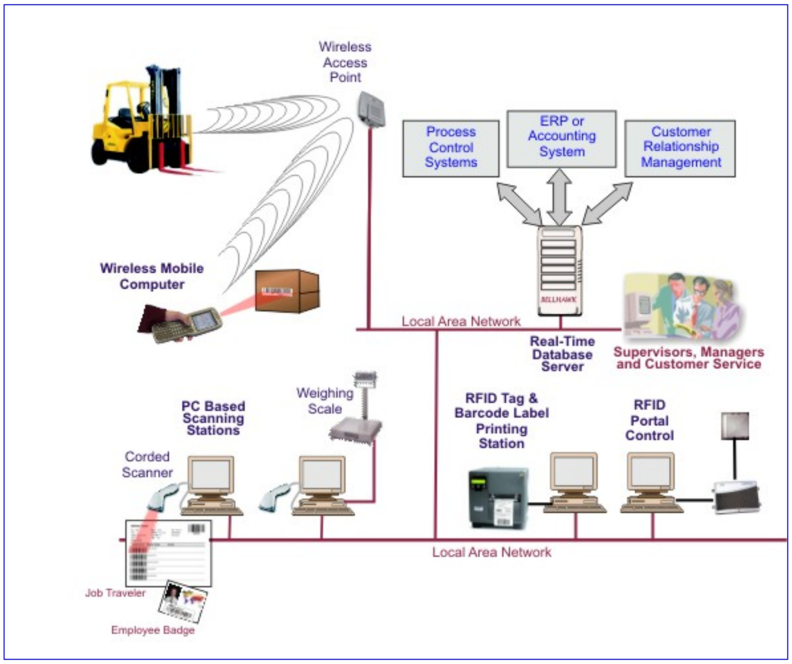
Deployment Diagrams: Examples



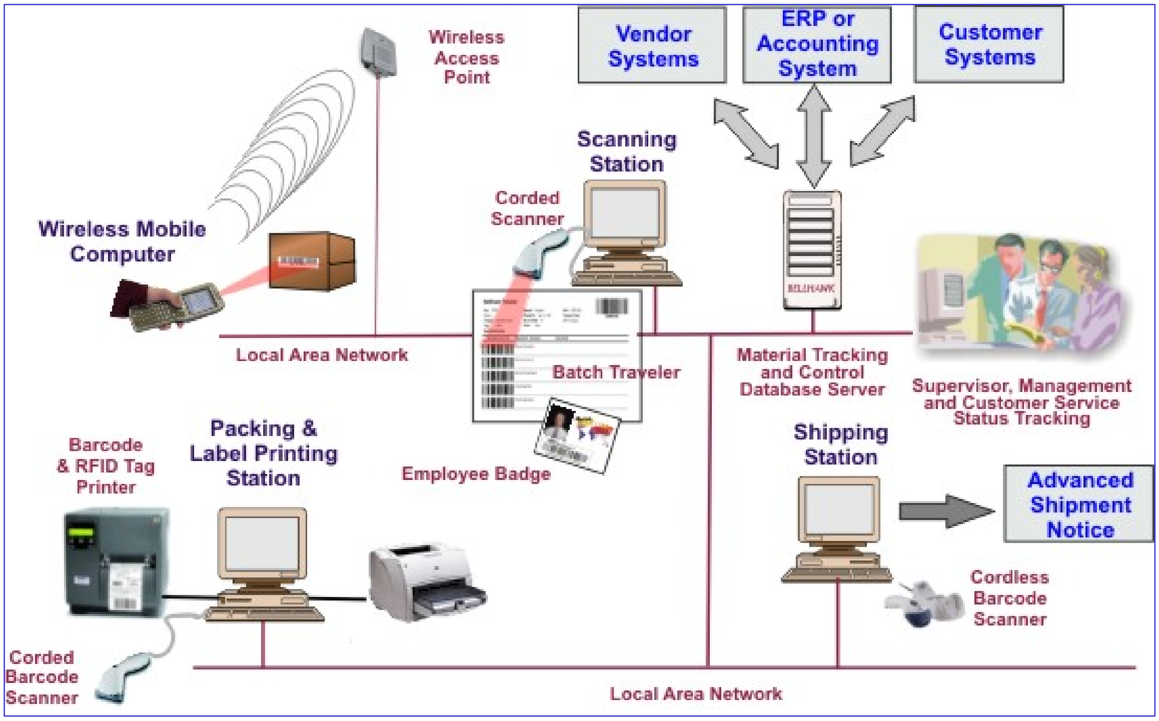
Deployment Diagrams: Examples



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