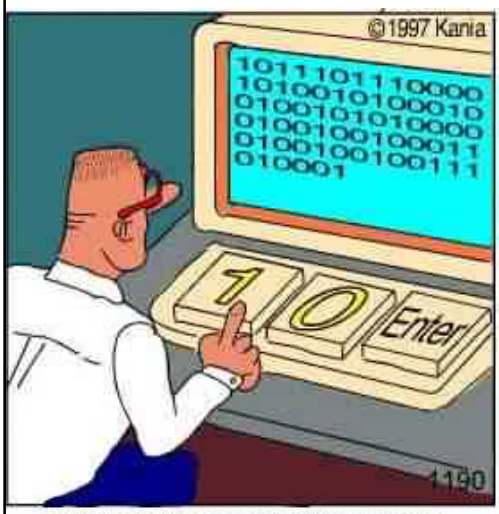


Real programmers code in binary.

Software Visualization

Software Architectures

Visualizing Software Architectures

- What is Software Architecture
- Some Diagrams on the Web
- Object Oriented
  - UML
  - Class Blueprint

(Software) Architecture

- Definition:
  - „Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.“

IEEE recommended practice for architecture description, IEEE Standard 1471, 2000.

Software Architecture

Issues:

- gross organization and global control structure;
- protocols for communication, synchronization, and data access;
- assignment of functionality to design elements;
- physical distribution;
- composition of design elements;
- scaling and performance;
- and selection among design alternatives.

An Introduction to Software Architecture, David Garlan, and Mary Shaw
In V. Ambriola and G. Tortora (ed.), *Advances in Software Engineering and Knowledge Engineering,* Series on Software Engineering and Knowledge Engineering, Vol 2, World Scientific Publishing Company, Singapore, pp. 1-39, 1993.

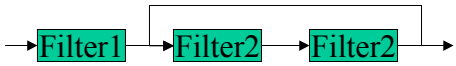
Software Architectures

Specification Method	**Relation**
• Functional Decomposition	⇒ Uses
• Data Flow	⇒ Becomes
• Data Structure	⇒ Is composed of
• Programming calculus	⇒ Constructs by proof (execution)

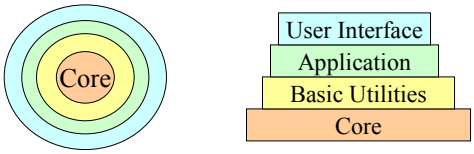
See [G.D. Bergland, „A Guided Tour of Program Design Methodologies“, IEEE Computer, October,1981]

Some Familiar Architectures

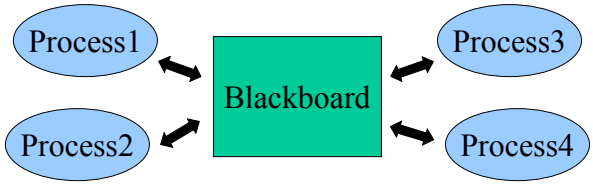
- Pipes and Filters



- Layered Systems



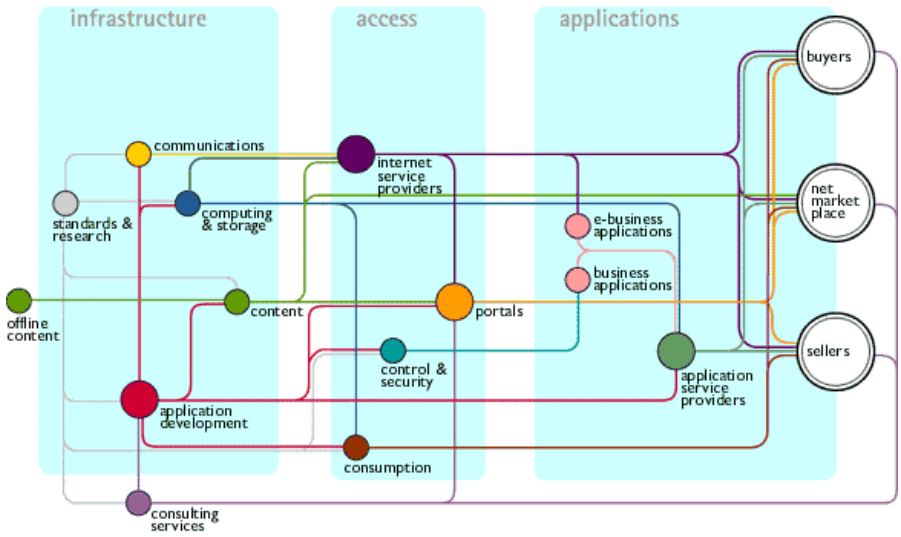
- Blackboard-driven (multiple units, read&write)

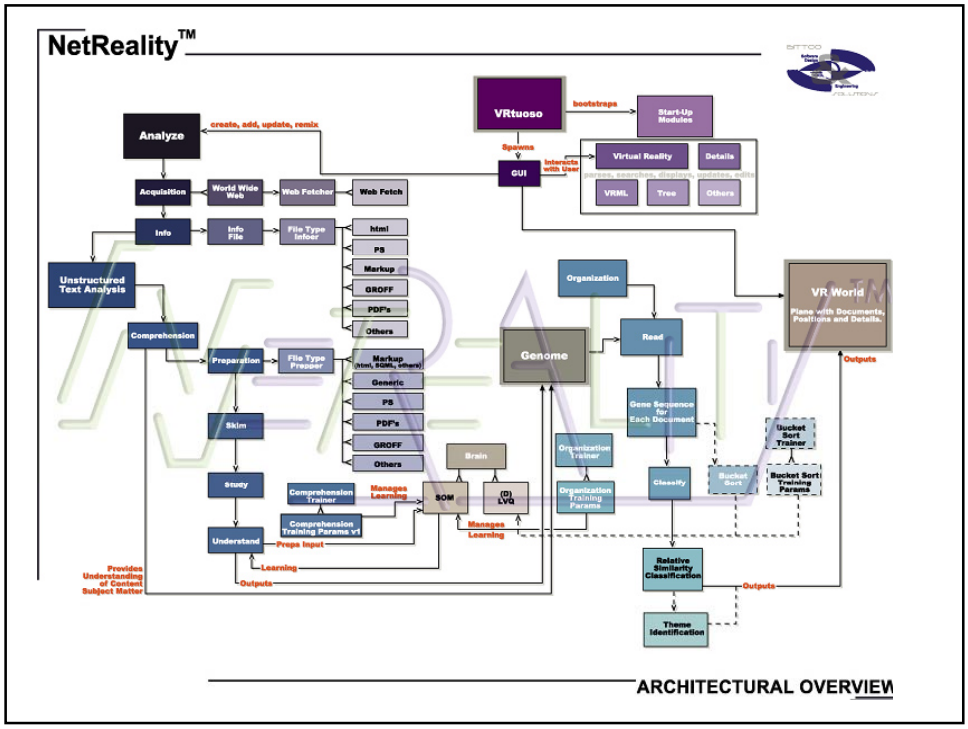


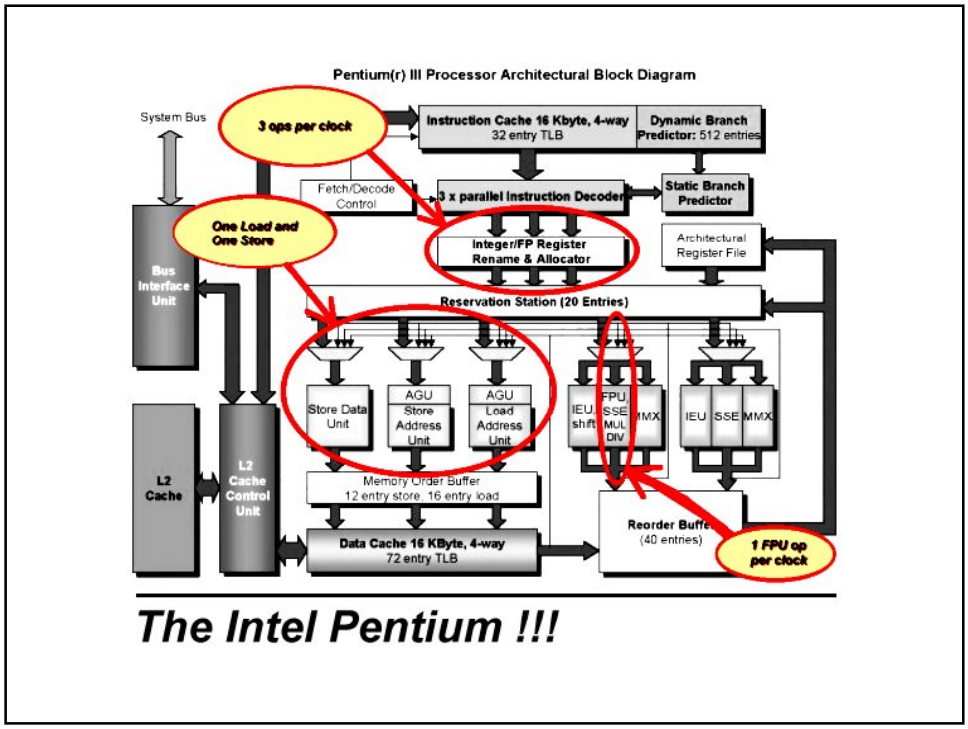
Some Architecture Diagrams found on the Internet

- IT Architectures:
  - include soft- and hardware and users
- Collected by Henk Koning
- Mostly freestyle graphics

See also [**Practical Guidelines for Readability of IT-architecture Diagrams**, Henk Koning, Claire Dormann, Hans van Vliet, SIGDOC2002]




NetReality™
VRtuoso
bootstraps
Start-Up Modules
Analyze
create, add, update, remix
Spawns
GUI
Interacts with User
Virtual Reality
Details
VRML
Tree
Others
Acquisition
World Wide Web
Web Fetcher
Web Fetch
Info
Info File
File Type Infoer
html
PS
Markup
GROFF
PDFs
Others
Unstructured Text Analysis
Comprehension
Preparation
File Type Prepper
Markup (html, SGML, others)
Generic
PS
PDFs
GROFF
Others
Skim
Study
Understand
Comprehension Trainer
Manages Learning
Comprehension Training Params vt
Preps Input
Learning
Outputs
Provides Understanding of Content Subject Matter
Brain
SOM
(D) LVQ
Manages Learning
Organization
Genome
Read
Organization Trainer
Organization Training Params
Gene Sequence for Each Document
Cleanify
Bucket Sort
Bucket Sort Trainer
Bucket Sort Training Params
VR World
Plane with Documents, Positions and Details.
Outputs
Relative Similarity Classification
Outputs
Theme Identification
ARCHITECTURAL OVERVIEW


Pentium(r) III Processor Architectural Block Diagram
System Bus
3 ops per clock
Instruction Cache 16 Kbyte, 4-way
32 entry TLB
Dynamic Branch Predictor: 512 entries
Fetch/Decode Control
3 x parallel Instruction Decoder
Static Branch Predictor
One Load and One Store
Integer/FP Register Rename & Allocator
Architectural Register File
Bus Interface Unit
Reservation Station (20 Entries)
Store Data Unit
AGU Store Address Unit
AGU Load Address Unit
IEU shift
FPU SSE MUL DIV MMX
IEU SSE MMX
L2 Cache
L2 Cache Control Unit
Memory Order Buffer
12 entry store 16 entry load
Data Cache 16 KByte, 4-way
72 entry TLB
Reorder Buffer
(40 entries)
1 FPU op per clock
The Intel Pentium !!!

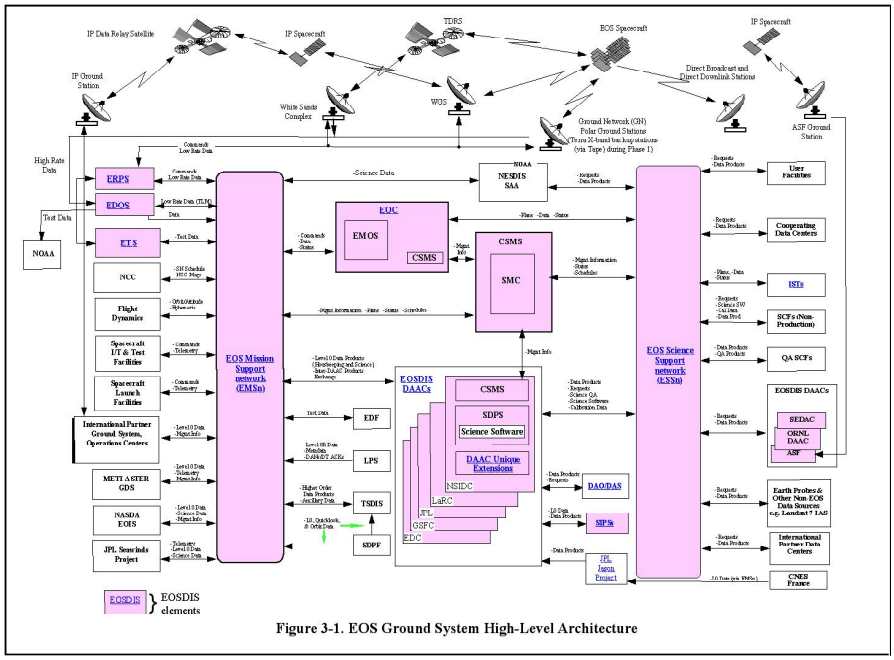
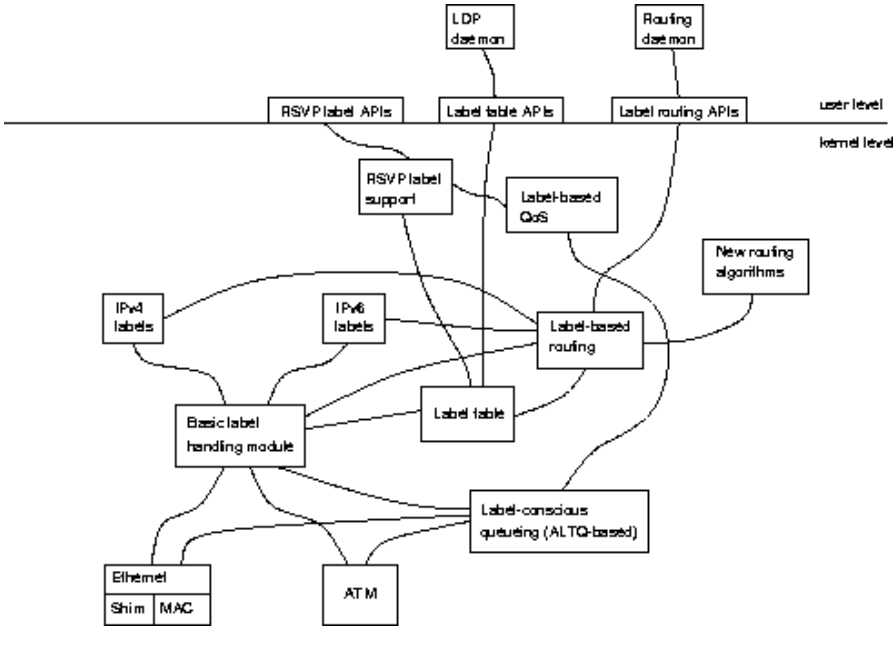
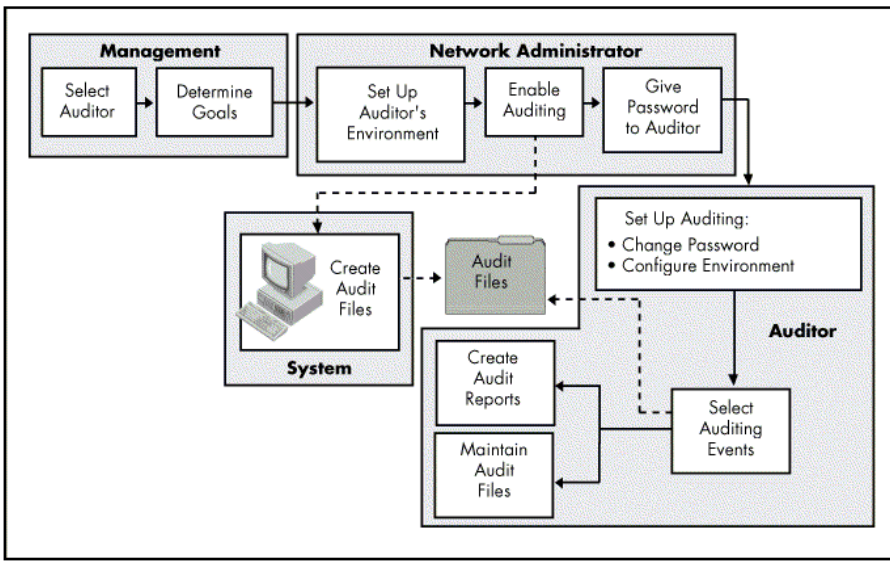
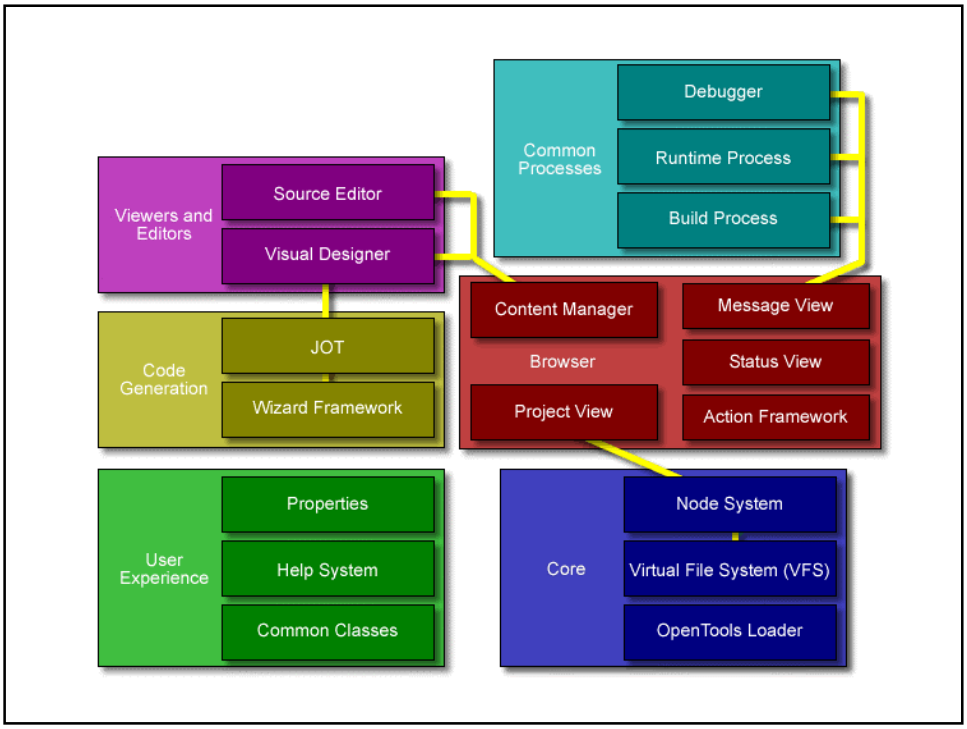
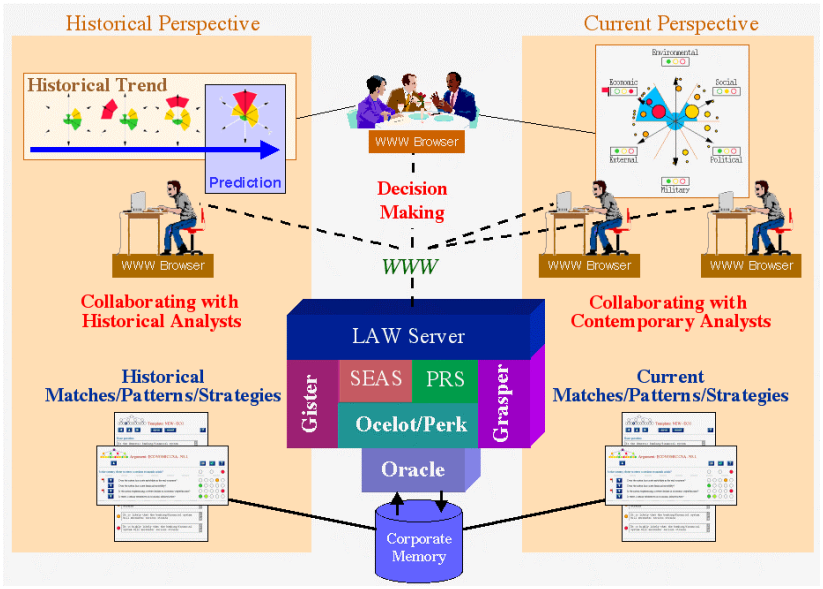


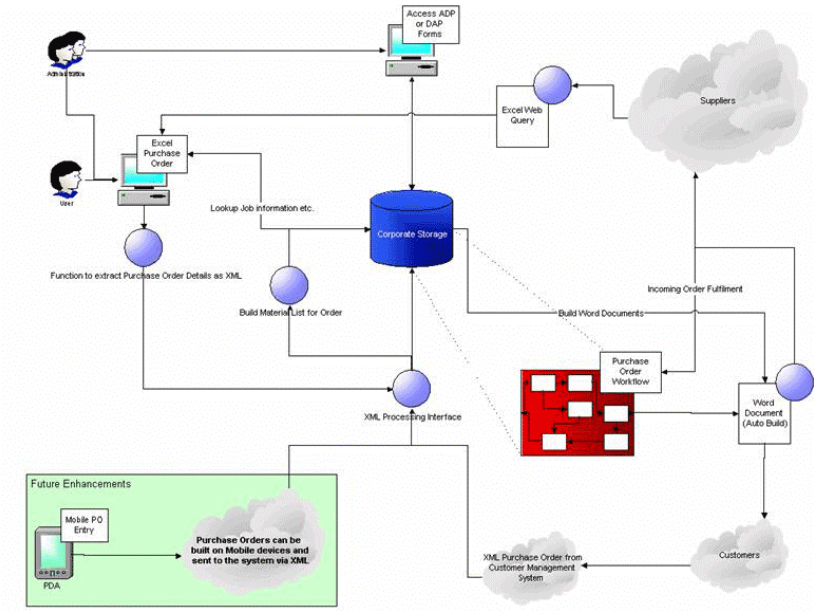
Figure 3-1. EOS Ground System High-Level Architecture

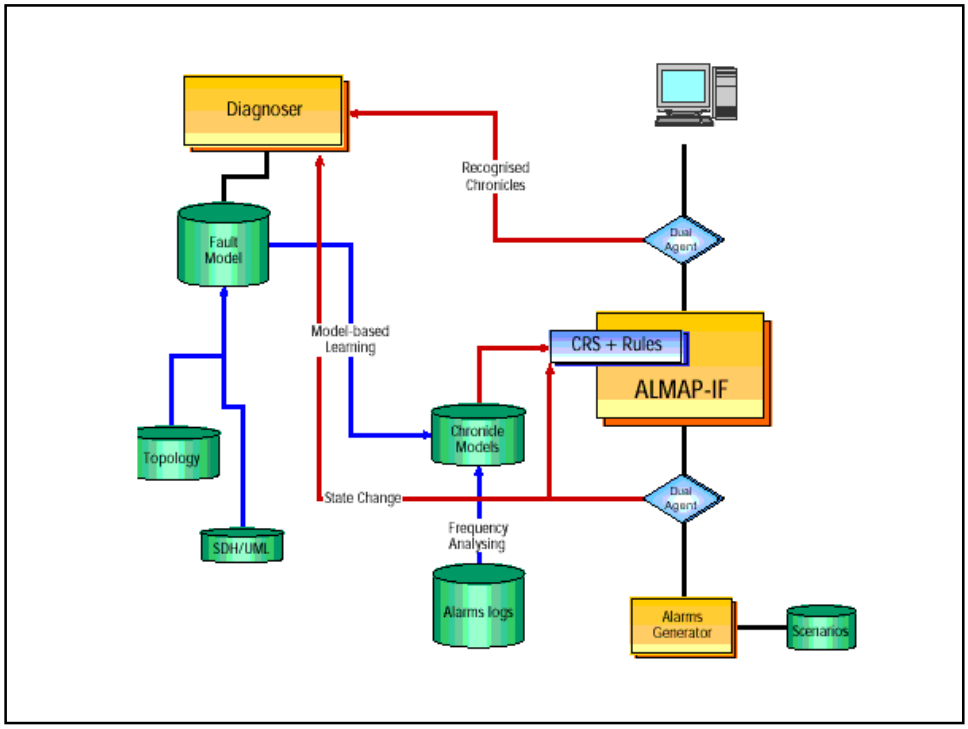


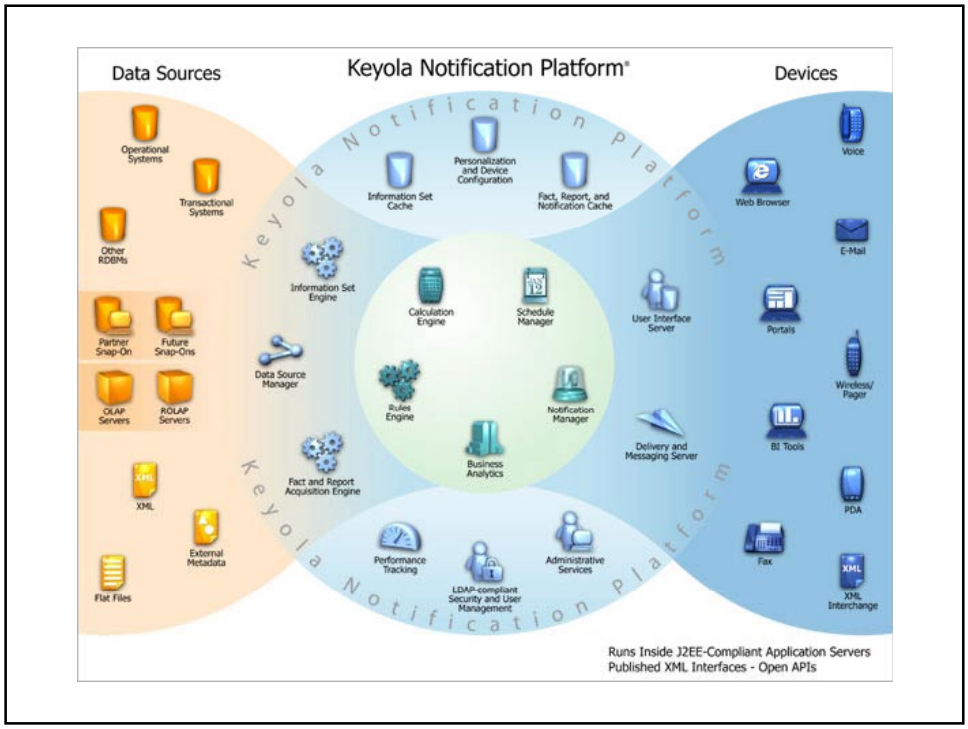

Management
Select Auditor
Determine Goals
Network Administrator
Set Up Auditor's Environment
Enable Auditing
Give Password to Auditor
Create Audit Files
System
Audit Files
Set Up Auditing:
• Change Password
• Configure Environment
Auditor
Create Audit Reports
Maintain Audit Files
Select Auditing Events

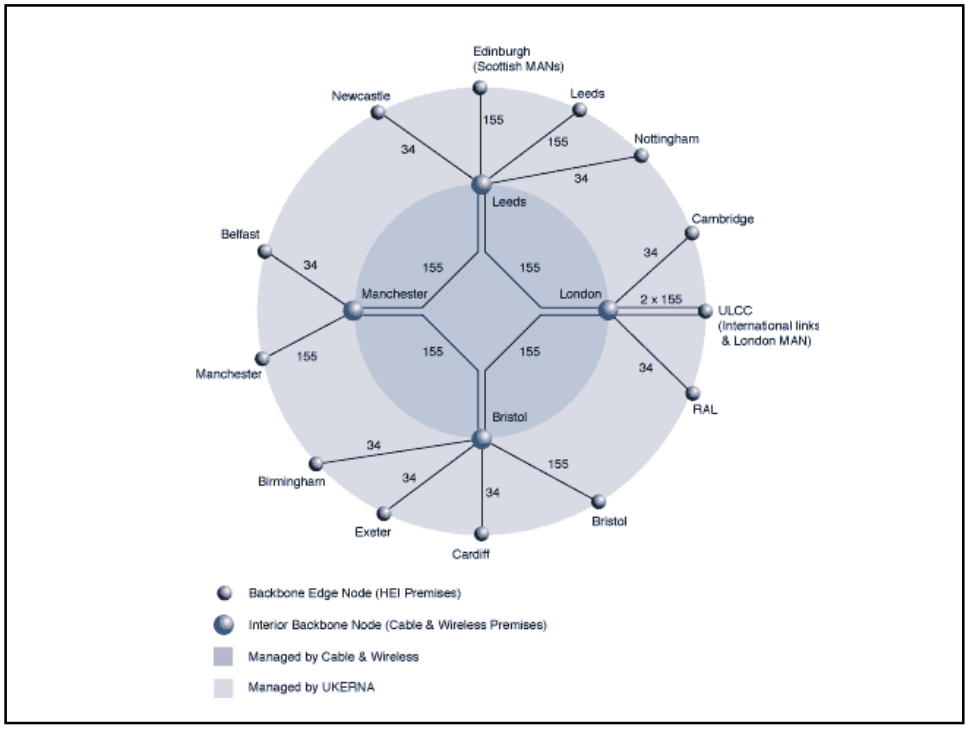

Debugger
Common Processes
Runtime Process
Build Process
Viewers and Editors
Source Editor
Visual Designer
Content Manager
Message View
Browser
Status View
Code Generation
JOT
Wizard Framework
Project View
Action Framework
Properties
User Experience
Help System
Common Classes
Node System
Core
Virtual File System (VFS)
OpenTools Loader

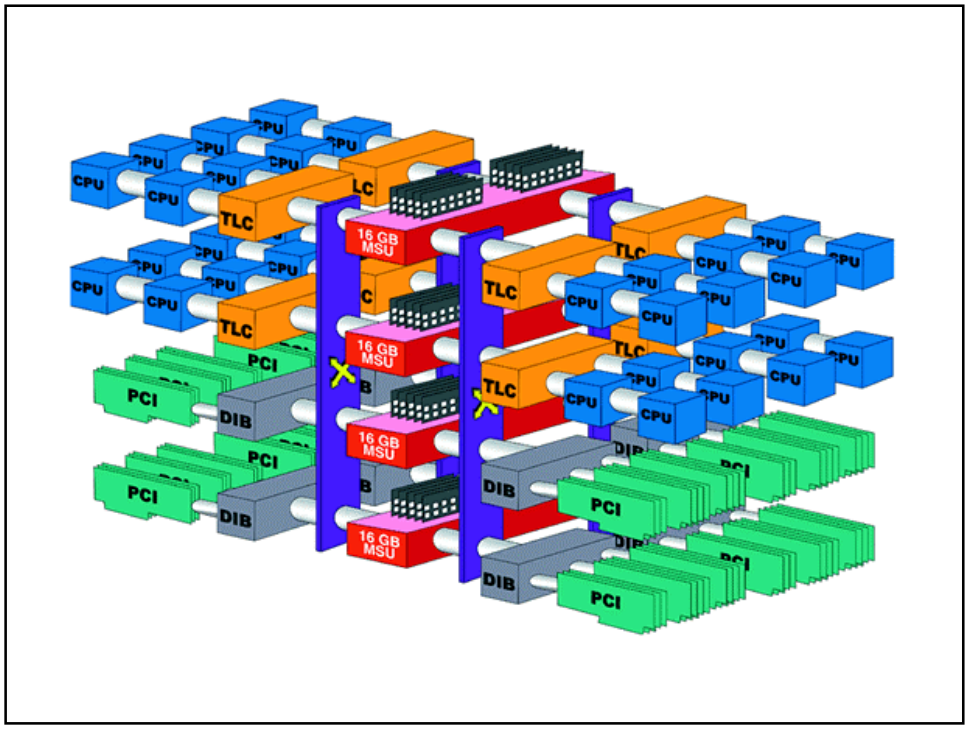

Historical Perspective
Current Perspective
Historical Trend
Prediction
WWW Browser
Decision Making
WWW
WWW Browser
WWW Browser
WWW Browser
Collaborating with Historical Analysts
Collaborating with Contemporary Analysts
LAW Server
Gister
SEAS
PRS
Ocelot/Perk
Grasper
Oracle
Historical Matches/Patterns/Strategies
Current Matches/Patterns/Strategies
Corporate Memory

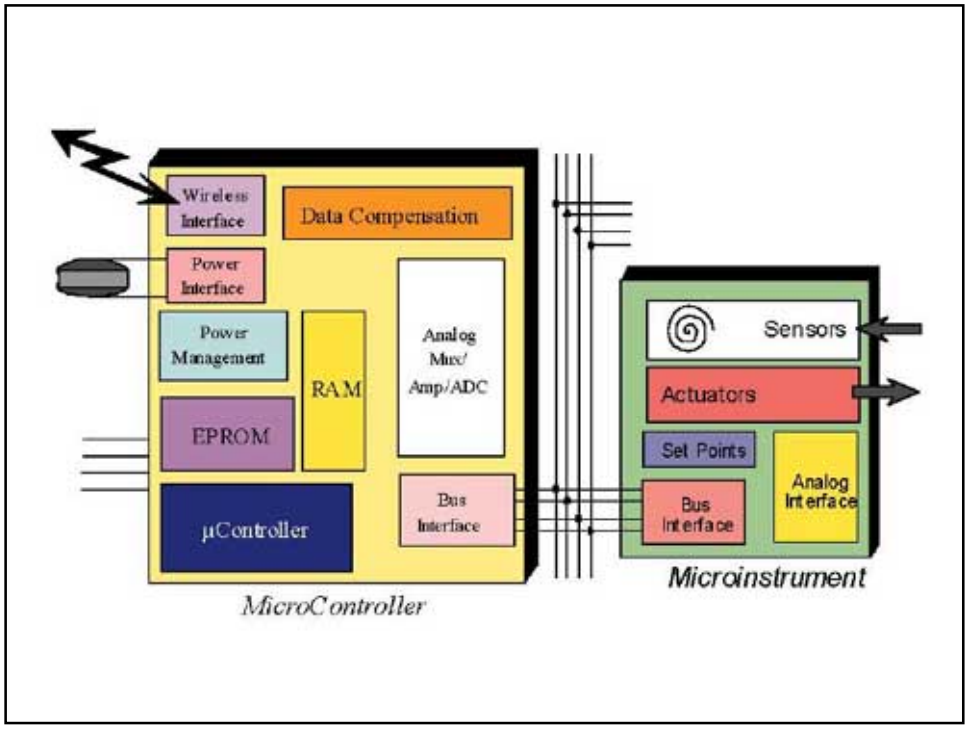

Access ADP or DAP Forms
Excel Web Query
Suppliers
Excel Purchase Order
Lookup Job information etc.
Corporate Storage
Function to extract Purchase Order Details as XML
Build Material List for Order
Incoming Order Fulfilment
Build Word Documents
Purchase Order Workflow
Word Document (Auto Build)
XML Processing Interface
Future Enhancements
Mobile PO Entry
PDA
Purchase Orders can be built on Mobile devices and sent to the system via XML
XML Purchase Order from Customer Management System
Customers

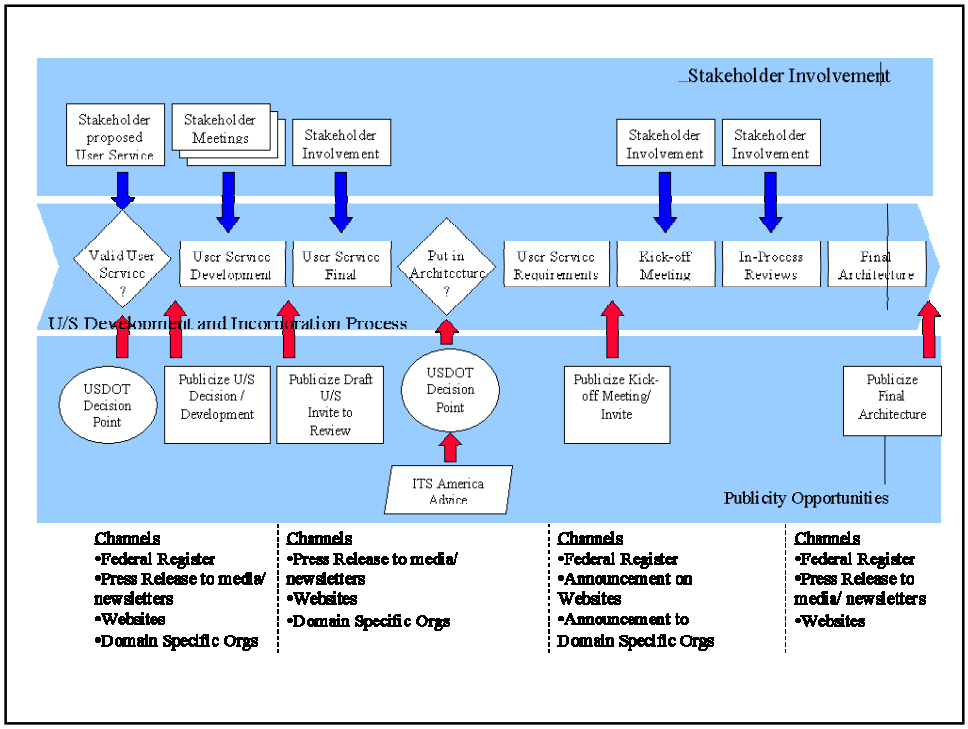

Diagnoser
Recognised Chronicles
Fault Model
Dual Agent
Model-based Learning
CRS + Rules
ALMAP-IF
Chronicle Models
Topology
State Change
Dual Agent
Frequency Analysing
SDH/UML
Alarms logs
Alarms Generator
Scenarios

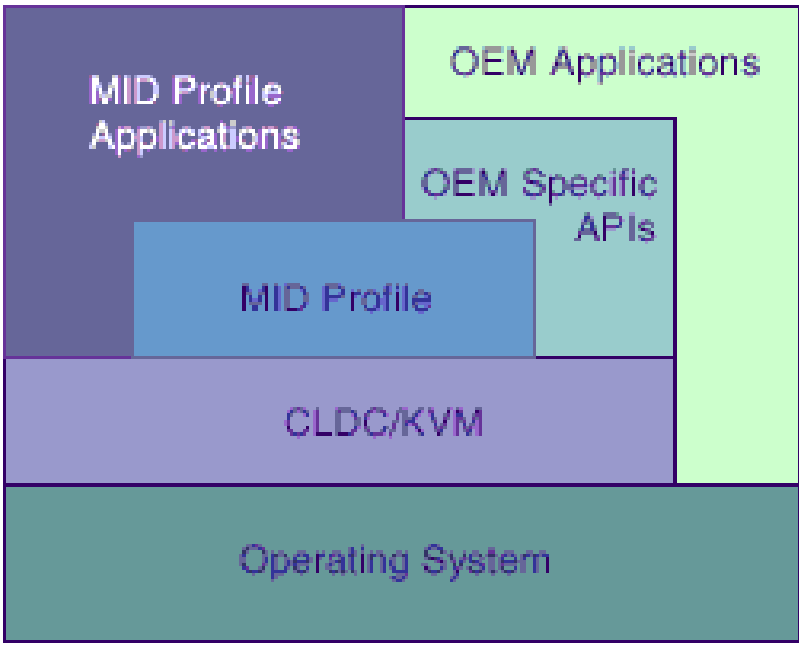

Data Sources
Keyola Notification Platform®
Devices
Operational Systems
Transactional Systems
Other RDBMs
Partner Snap-On
Future Snap-Ons
OLAP Servers
ROLAP Servers
XML
External Metadata
Flat Files
Information Set Cache
Personalization and Device Configuration
Fact, Report, and Notification Cache
Information Set Engine
Calculation Engine
Schedule Manager
Data Source Manager
Rules Engine
Notification Manager
Business Analytics
Fact and Report Acquisition Engine
Performance Tracking
LDAP-compliant Security and User Management
Administrative Services
User Interface Server
Delivery and Messaging Server
Voice
Web Browser
E-Mail
Portals
Wireless/ Pager
BI Tools
PDA
Fax
XML Interchange
Runs Inside J2EE-Compliant Application Servers
Published XML Interfaces - Open APIs

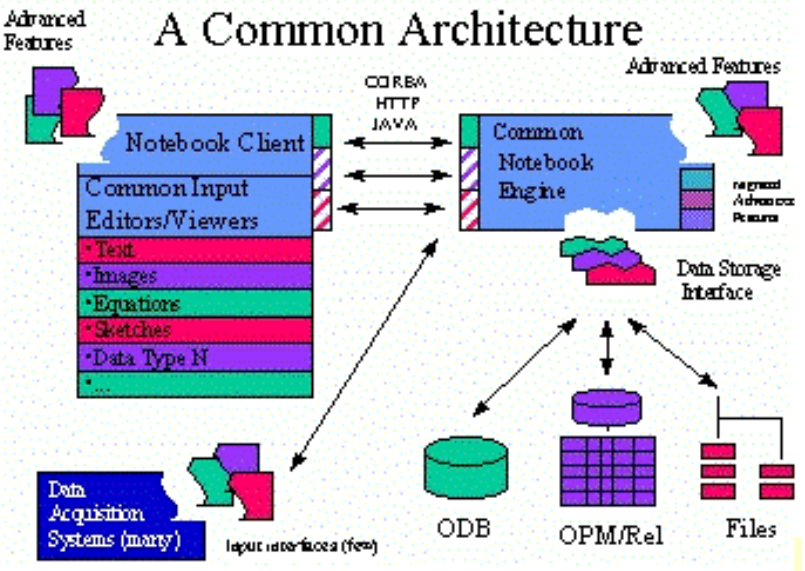

Edinburgh
(Scottish MANs)
Newcastle
Leeds
Nottingham
Leeds
Cambridge
Belfast
Manchester
London
2 x 155
ULCC
(International links
& London MAN)
Manchester
RAL
Bristol
Birmingham
Bristol
Exeter
Cardiff
155
34
Backbone Edge Node (HEI Premises)
Interior Backbone Node (Cable & Wireless Premises)
Managed by Cable & Wireless
Managed by UKERNA

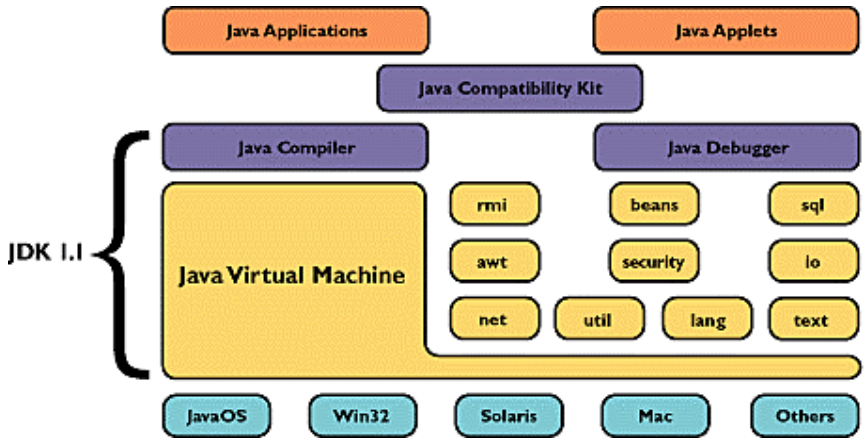

CPU
TLC
16 GB
MSU
PCI
DIB

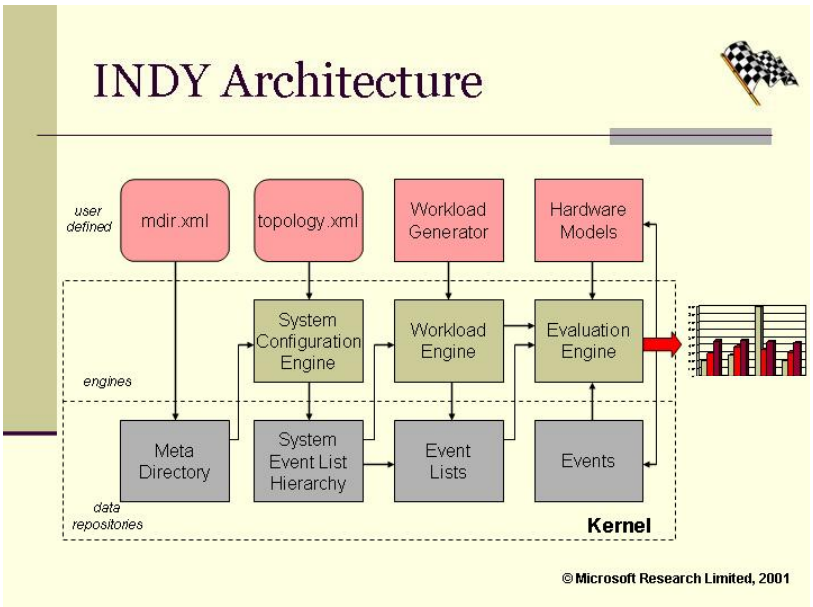

Wireless Interface
Data Compensation
Power Interface
Power Management
RAM
Analog Mux/ Amp/ADC
EPROM
µController
Bus Interface
MicroController
Sensors
Actuators
Set Points
Analog Interface
Bus Interface
Microinstrument

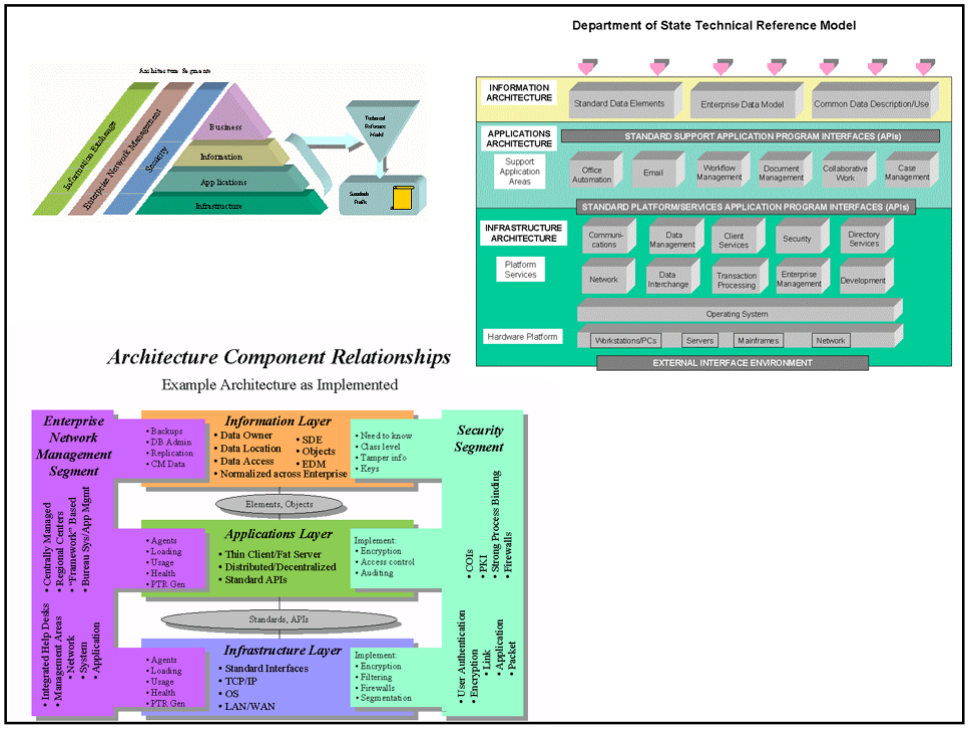

Stakeholder Involvement
Stakeholder proposed User Service
Stakeholder Meetings
Stakeholder Involvement
Stakeholder Involvement
Stakeholder Involvement
Valid User Service ?
User Service Development
User Service Final
Put in Architecture ?
User Service Requirements
Kick-off Meeting
In-Process Reviews
Final Architecture
U/S Development and Incorporation Process
USDOT Decision Point
Publicize U/S Decision / Development
Publicize Draft U/S Invite to Review
USDOT Decision Point
ITS America Advice
Publicize Kick-off Meeting/ Invite
Publicize Final Architecture
Publicity Opportunities
Channels
•Federal Register
•Press Release to media/ newsletters
•Websites
•Domain Specific Orgs
Channels
•Press Release to media/ newsletters
•Websites
•Domain Specific Orgs
Channels
•Federal Register
•Announcement on Websites
•Announcement to Domain Specific Orgs
Channels
•Federal Register
•Press Release to media/ newsletters
•Websites

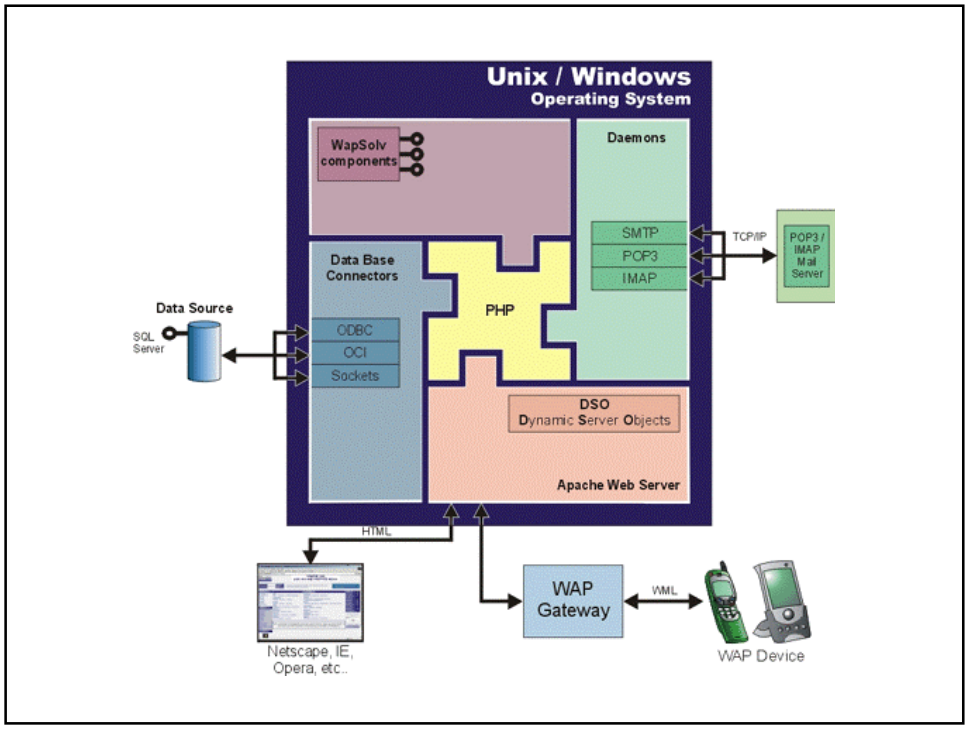

MID Profile Applications
OEM Applications
OEM Specific APIs
MID Profile
CLDC/KVM
Operating System

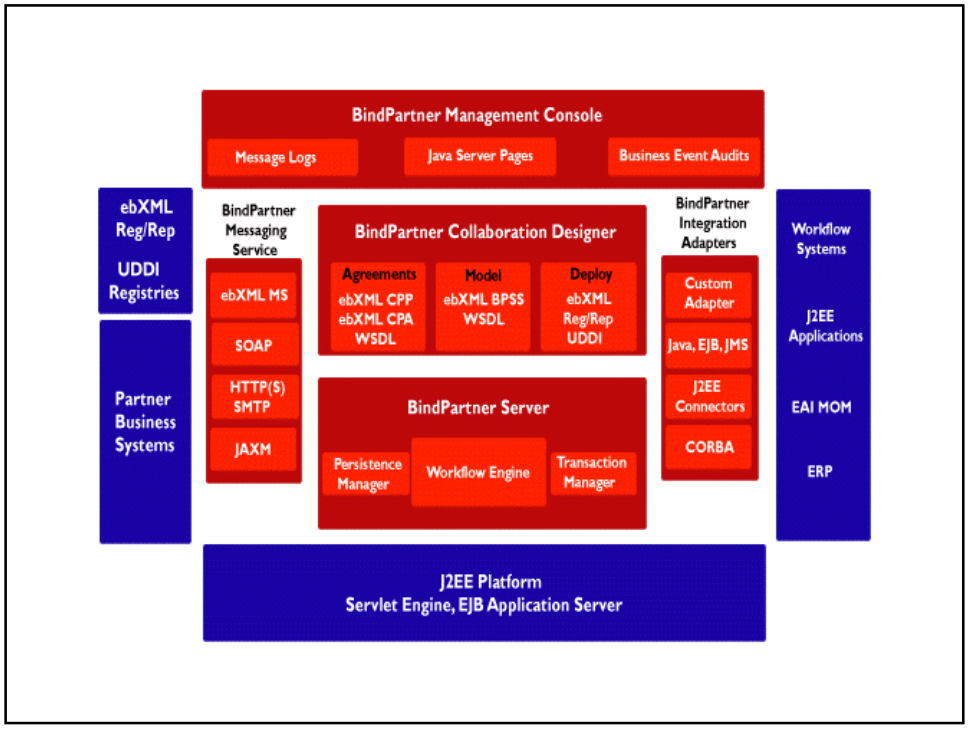

A Common Architecture
Advanced Features
CORBA
HTTP
JAVA
Notebook Client
Common Input Editors/Viewers
·Text
·Images
·Equations
·Sketches
·Data Type N
·...
Common Notebook Engine
Advanced Features
Data Storage Interface
Data Acquisition Systems (many)
ODB
OPM/Rel
Files

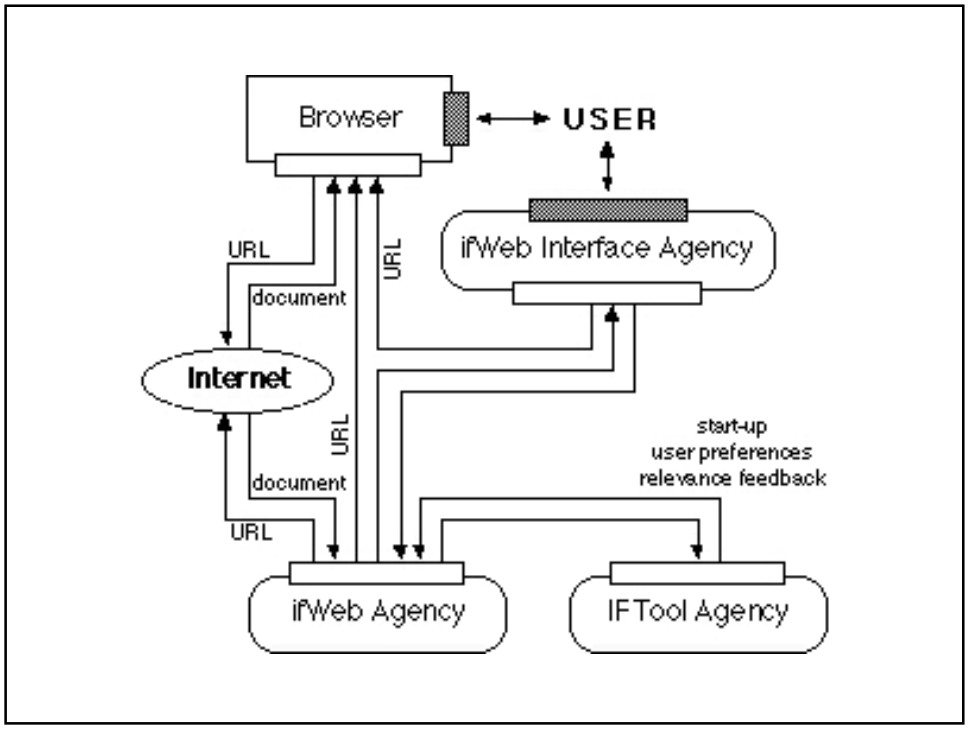

Java Applications
Java Applets
Java Compatibility Kit
Java Compiler
Java Debugger
JDK 1.1
Java Virtual Machine
rmi
beans
sql
awt
security
io
net
util
lang
text
JavaOS
Win32
Solaris
Mac
Others

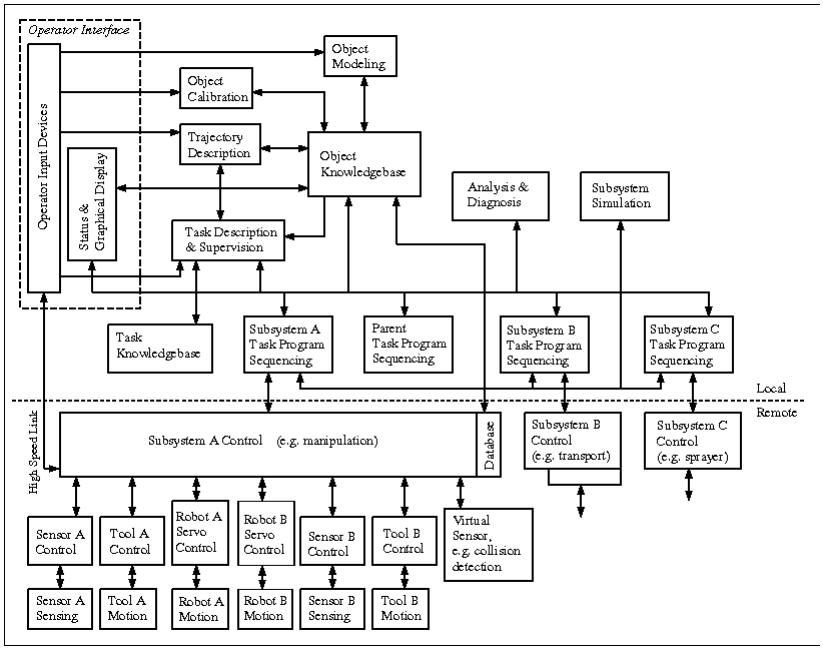

INDY Architecture
user defined
mdir.xml
topology.xml
Workload Generator
Hardware Models
System Configuration Engine
Workload Engine
Evaluation Engine
engines
Meta Directory
System Event List Hierarchy
Event Lists
Events
data repositories
Kernel
© Microsoft Research Limited, 2001

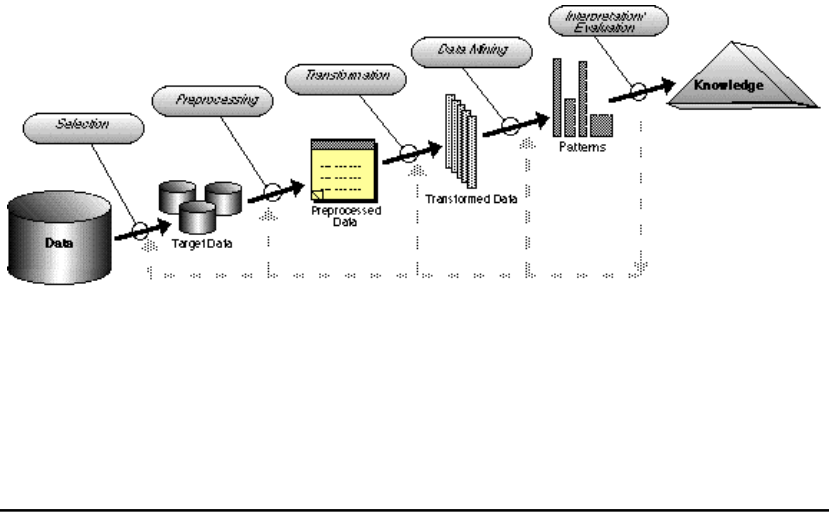

Department of State Technical Reference Model
INFORMATION ARCHITECTURE
Standard Data Elements
Enterprise Data Model
Common Data Description/Use
APPLICATIONS ARCHITECTURE
STANDARD SUPPORT APPLICATION PROGRAM INTERFACES (APIs)
Support Application Areas
Office Automation
Email
Workflow Management
Document Management
Collaborative Work
Case Management
STANDARD PLATFORM/SERVICES APPLICATION PROGRAM INTERFACES (APIs)
INFRASTRUCTURE ARCHITECTURE
Communi-cations
Data Management
Client Services
Security
Directory Services
Platform Services
Network
Data Interchange
Transaction Processing
Enterprise Management
Development
Operating System
Hardware Platform
Workstations/PCs
Servers
Mainframes
Network
EXTERNAL INTERFACE ENVIRONMENT
Architecture Segments
Business
Information
Applications
Infrastructure
Information Exchange
Enterprise Network Management
Security
Standards Profile
Architecture Component Relationships
Example Architecture as Implemented
Enterprise Network Management Segment
• Centrally Managed
• Regional Centers
• "Framework" Based
• Bureau Sys/App Mgmt
• Integrated Help Desks
• Management Areas
• Network
• System
• Application
Information Layer
• Data Owner
• Data Location
• Data Access
• Normalized across Enterprise
• SDE
• Objects
• EDM
• Backups
• DB Admin
• Replication
• CM Data
• Need to know
• Class level
• Tamper info
• Keys
Elements, Objects
Applications Layer
• Thin Client/Fat Server
• Distributed/Decentralized
• Standard APIs
• Agents
• Loading
• Usage
• Health
• PTR Gen
Implement:
• Encryption
• Access control
• Auditing
Standards, APIs
Infrastructure Layer
• Standard Interfaces
• TCP/IP
• OS
• LAN/WAN
• Agents
• Loading
• Usage
• Health
• PTR Gen
Implement:
• Encryption
• Filtering
• Firewalls
• Segmentation
Security Segment
• COIs
• PKI
• Strong Process Binding
• Firewalls
• User Authentication
• Encryption
• Link
• Application
• Packet

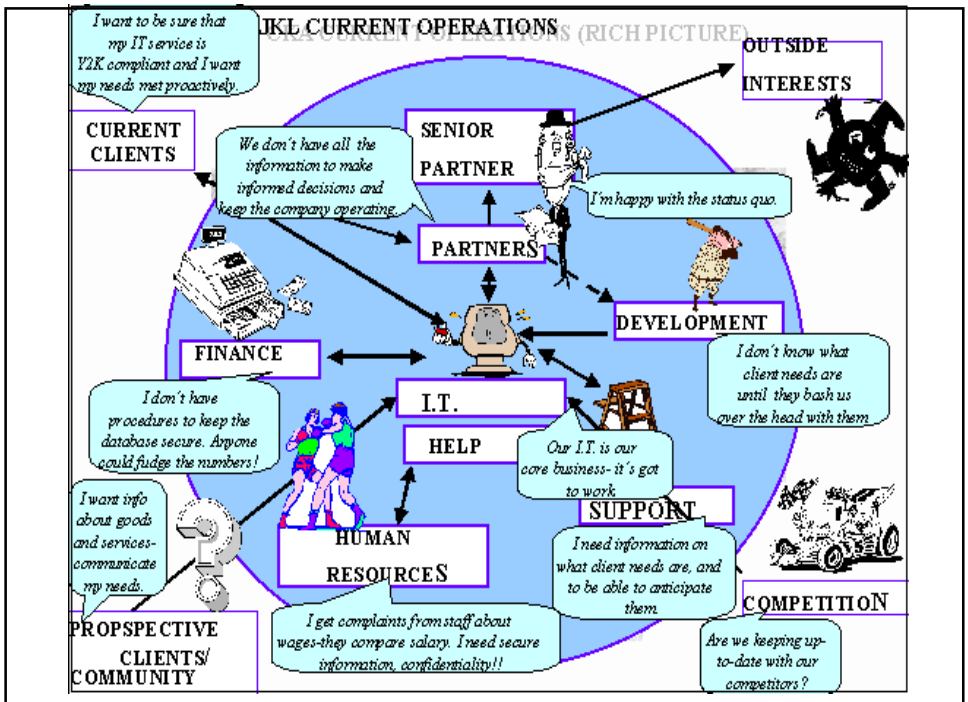

Unix / Windows
Operating System
WapSolv components
Daemons
SMTP
POP3
IMAP
TCP/IP
POP3 / IMAP Mail Server
Data Base Connectors
PHP
Data Source
SQL Server
ODBC
OCI
Sockets
DSO
Dynamic Server Objects
Apache Web Server
HTML
WAP Gateway
WML
Netscape, IE, Opera, etc..
WAP Device

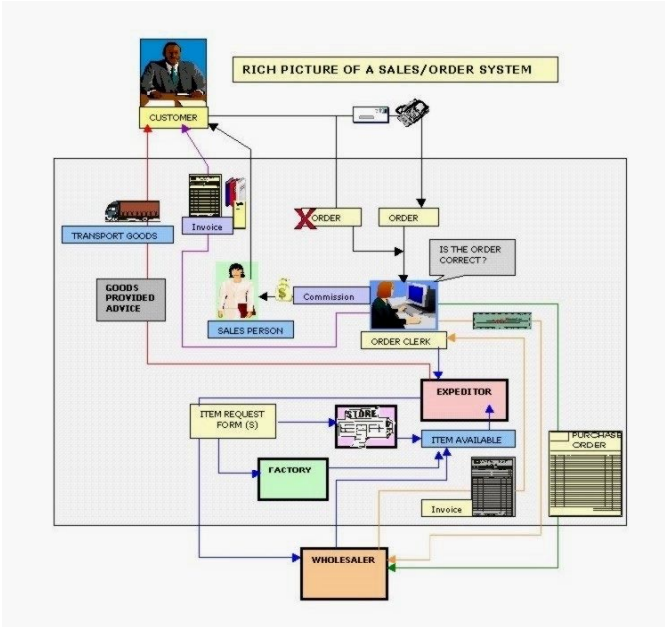

BindPartner Management Console
Message Logs
Java Server Pages
Business Event Audits
ebXML Reg/Rep
UDDI Registries
Partner Business Systems
BindPartner Messaging Service
ebXML MS
SOAP
HTTP(S) SMTP
JAXM
BindPartner Collaboration Designer
Agreements
ebXML CPP
ebXML CPA
WSDL
Model
ebXML BPSS
WSDL
Deploy
ebXML Reg/Rep
UDDI
BindPartner Server
Persistence Manager
Workflow Engine
Transaction Manager
BindPartner Integration Adapters
Custom Adapter
Java, EJB, JMS
J2EE Connectors
CORBA
Workflow Systems
J2EE Applications
EAI MOM
ERP
J2EE Platform
Servlet Engine, EJB Application Server

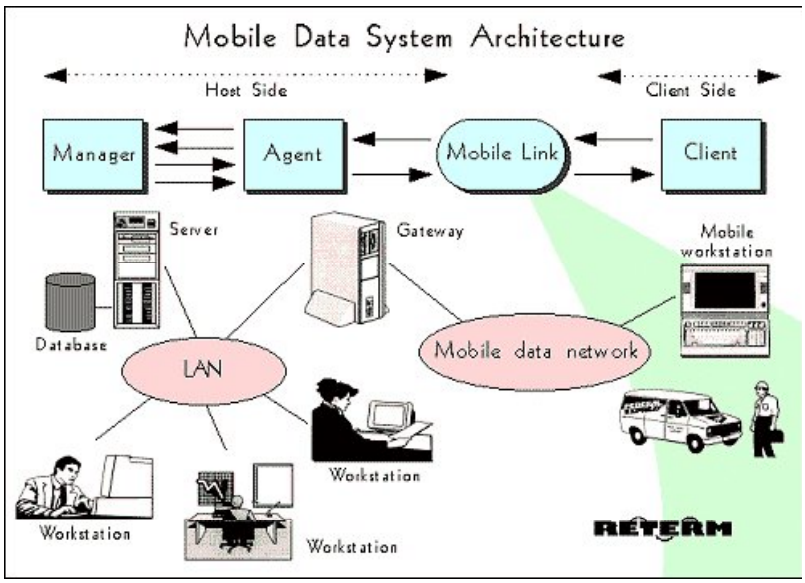

Browser
USER
ifWeb Interface Agency
URL
document
URL
Internet
URL
document
URL
start-up
user preferences
relevance feedback
ifWeb Agency
IF Tool Agency

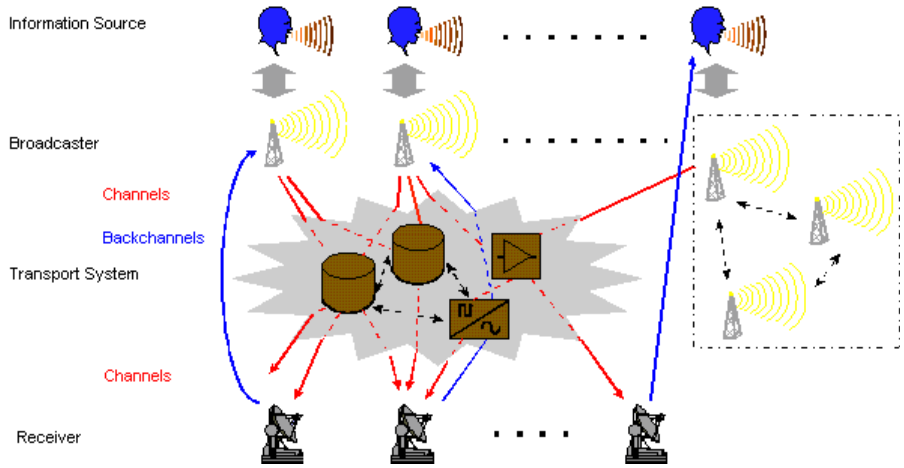

Operator Interface
Operator Input Devices
Status & Graphical Display
Object Modeling
Object Calibration
Trajectory Description
Object Knowledgebase
Task Description & Supervision
Analysis & Diagnosis
Subsystem Simulation
Task Knowledgebase
Subsystem A Task Program Sequencing
Parent Task Program Sequencing
Subsystem B Task Program Sequencing
Subsystem C Task Program Sequencing
Local
Remote
High Speed Link
Subsystem A Control (e.g. manipulation)
Database
Subsystem B Control (e.g. transport)
Subsystem C Control (e.g. sprayer)
Sensor A Control
Tool A Control
Robot A Servo Control
Robot B Servo Control
Sensor B Control
Tool B Control
Virtual Sensor, e.g. collision detection
Sensor A Sensing
Tool A Motion
Robot A Motion
Robot B Motion
Sensor B Sensing
Tool B Motion

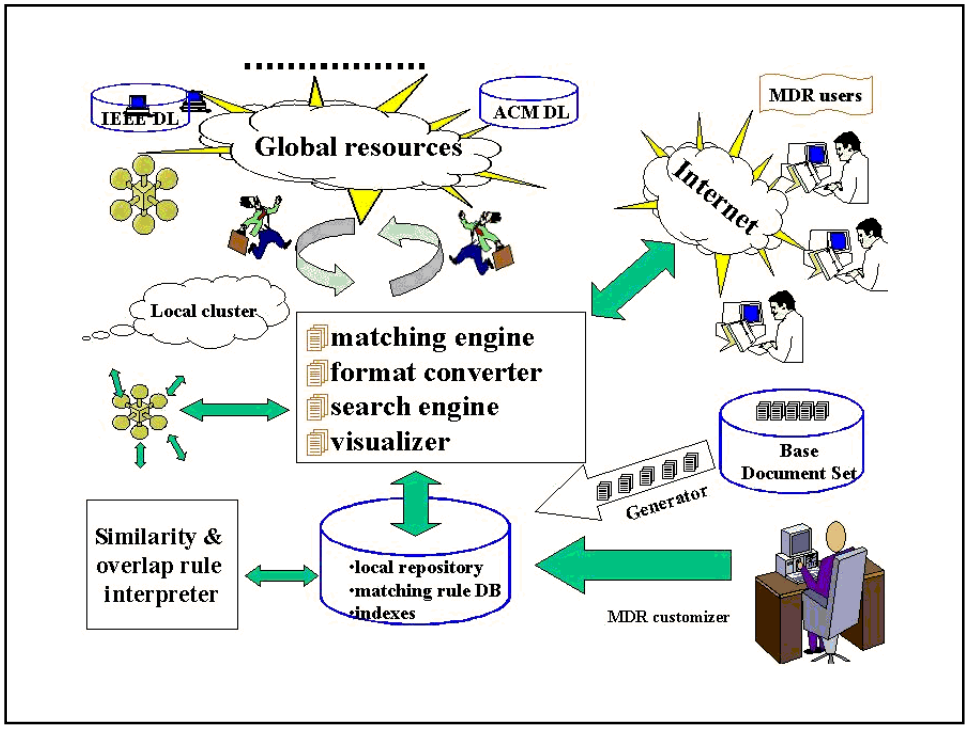

Interpretation / Evaluation
Data Mining
Transformation
Preprocessing
Selection
Knowledge
Patterns
Transformed Data
Preprocessed Data
Target Data
Data


JKL CURRENT OPERATIONS (RICH PICTURE)
I want to be sure that my IT service is Y2K compliant and I want my needs met proactively.
CURRENT CLIENTS
We don't have all the information to make informed decisions and keep the company operating.
SENIOR PARTNER
OUTSIDE INTERESTS
I'm happy with the status quo.
PARTNERS
DEVELOPMENT
FINANCE
I don't know what client needs are until they bash us over the head with them.
I don't have procedures to keep the database secure. Anyone could fudge the numbers!
I.T.
HELP
Our I.T. is our core business- it's got to work.
SUPPORT
I want info about goods and services- communicate my needs.
HUMAN RESOURCES
I need information on what client needs are, and to be able to anticipate them.
COMPETITION
PROPSPECTIVE CLIENTS/ COMMUNITY
I get complaints from staff about wages-they compare salary. I need secure information, confidentiality!!
Are we keeping up-to-date with our competitors?


RICH PICTURE OF A SALES/ORDER SYSTEM
CUSTOMER
TRANSPORT GOODS
Invoice
ORDER
ORDER
IS THE ORDER CORRECT?
GOODS PROVIDED ADVICE
Commission
SALES PERSON
ORDER CLERK
EXPEDITOR
ITEM REQUEST FORM (S)
STORE
ITEM AVAILABLE
PURCHASE ORDER
FACTORY
Invoice
WHOLESALER

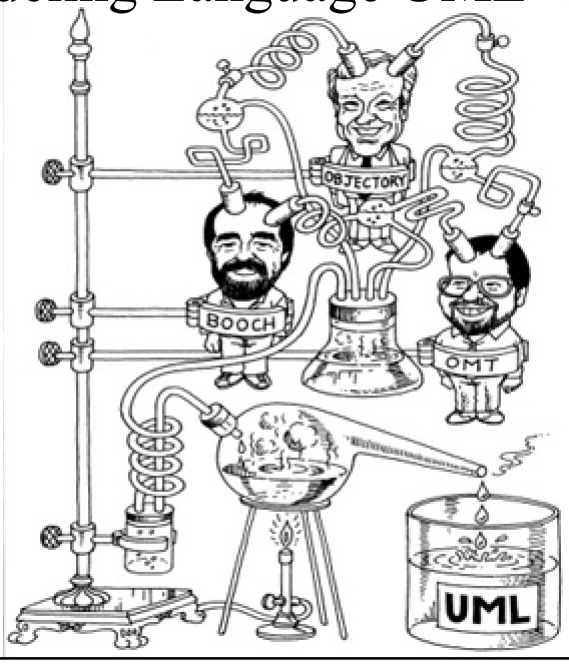

Mobile Data System Architecture
Host Side
Client Side
Manager
Agent
Mobile Link
Client
Server
Gateway
Mobile workstation
Database
LAN
Mobile data network
Workstation
Workstation
Workstation
RETERM


Information Source
Broadcaster
Channels
Backchannels
Transport System
Channels
Receiver



Universal Modeling Language UML

- A set of graphical notations for modeling software systems
- Combination of the methods of Booch, Rumbaugh und Jacobsen („Three Amigos“).
- Standardized by the Object Management Group (OMG)

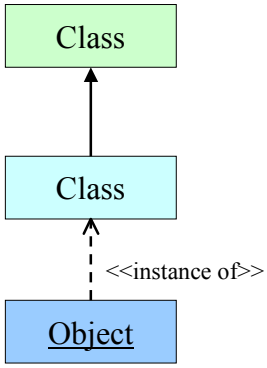


UML Diagrams

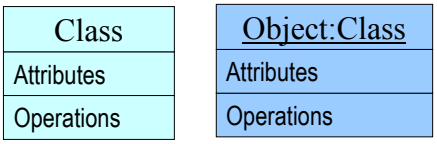
- use case diagram
- class diagram (including object diagram)
- behavior diagrams:
  - statechart diagram
  - activity diagram
- interaction diagrams:
  - sequence diagram
  - collaboration diagram
- implementation diagrams:
  - component diagram
  - deployment diagram
- model management diagrams:
  - packages, subsystems, and models

Graphical Notation

- We look here only at a small subset of UML

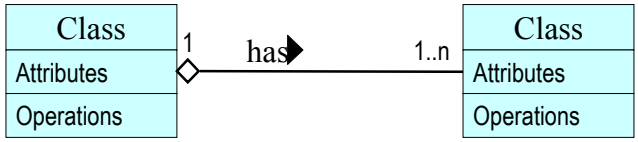


State and Behavior:



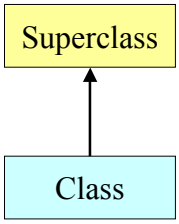
Graphical Notation

- Aggregation



Notation for Multiplicity: 1..n, n..k, 1..*, *

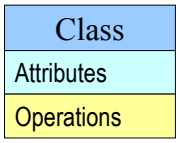
Classes



- Java Syntax:
  class *Class* **extends** *Superclass* **{** *Properties* **}**

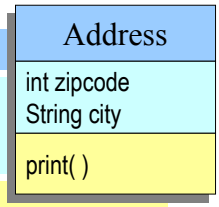
- Example:
  class Address { ... }
  class HomeAddress extends Address { ... }
  class POBoxAddress extends Address { ... }

Properties

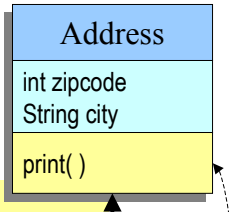


- Syntax: **class** *Class* **extends** *Superclass*
  { *Variable*
  and Method declarations **}**
- Example:

```
class Address
   {    int zipcode;
        String city;
        void print()
              { System.out.println(zipcode+" "+city);  }
   }
```

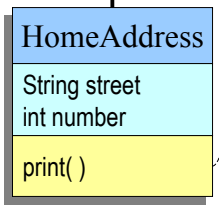


Method Overriding



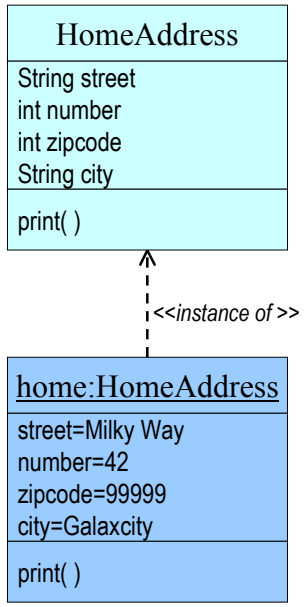

```
class Address    {   int zipcode;
                     String city;
                     void print()
                           { System.out.println(zipcode+" "+city);  }
                   }

class HomeAddress extends Address
   {      String street;
          int number;
          void print()
                 {  System.out.println(street+" "+number);
                    super.print();  }
   }
```

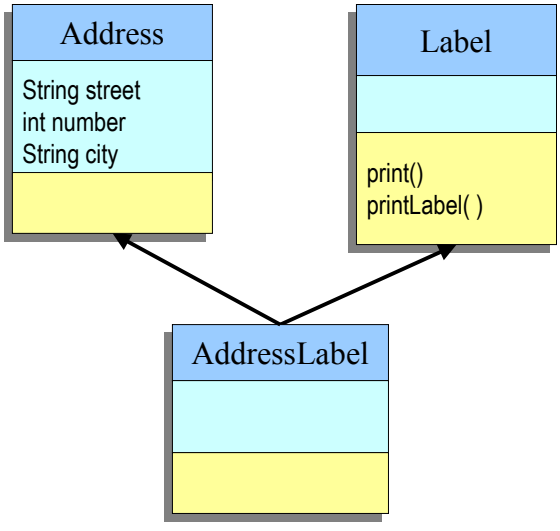



Call to a hidden method, method chaining

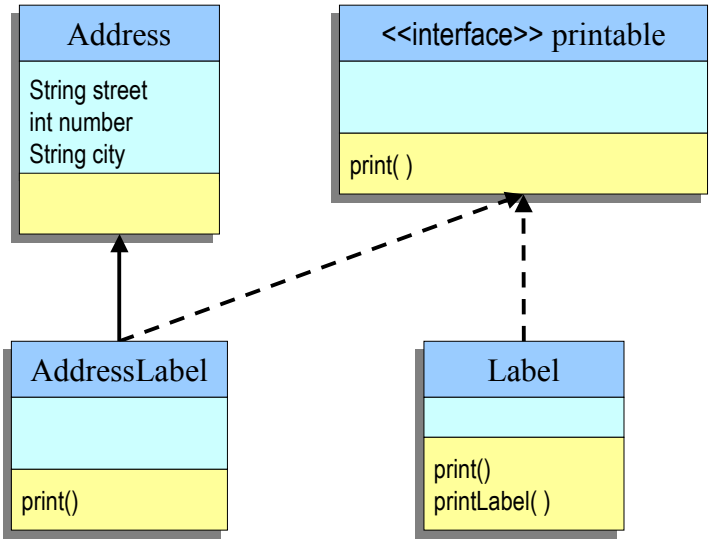
Objects



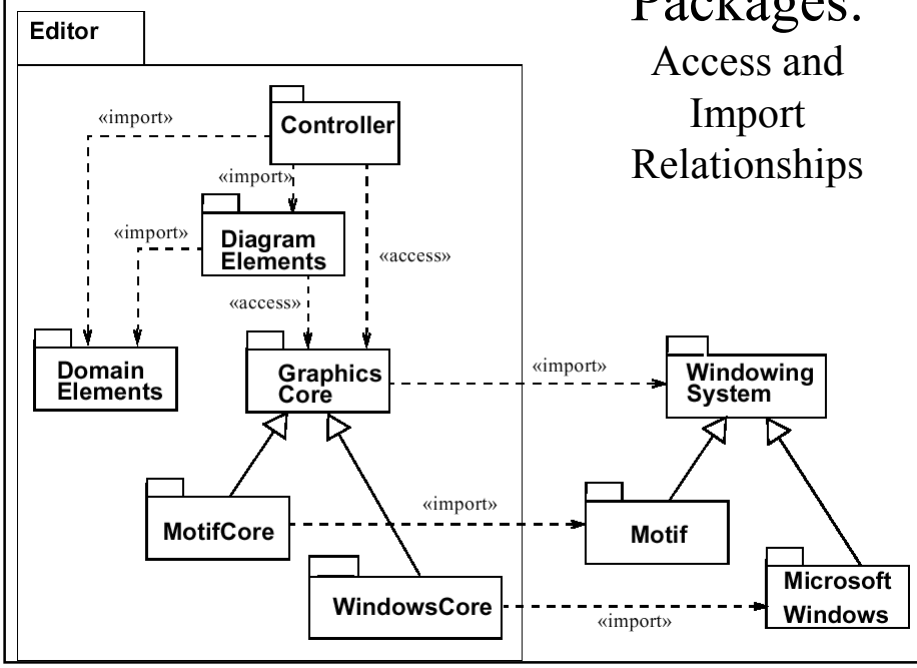
Multiple Inheritance



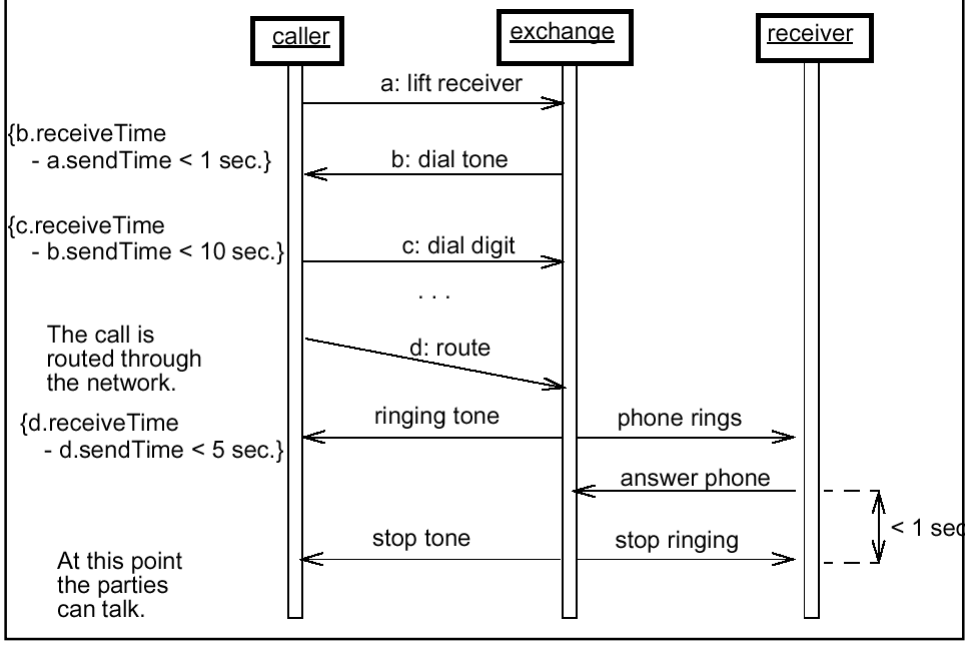
Interfaces



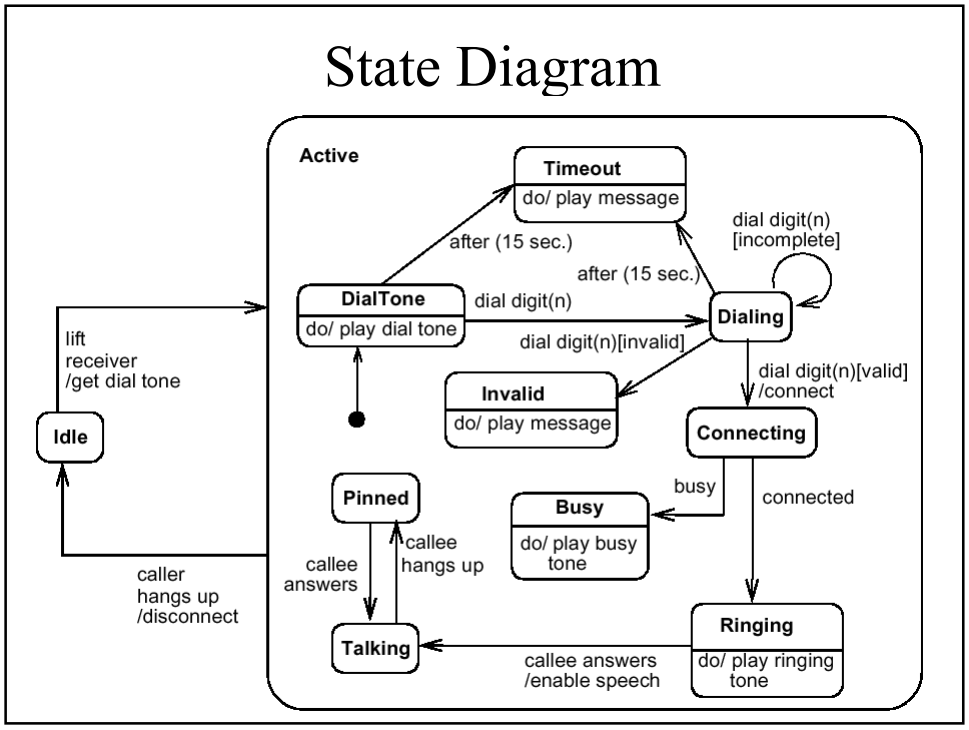
Packages:
Access and Import Relationships



Sequence Diagram



State Diagram

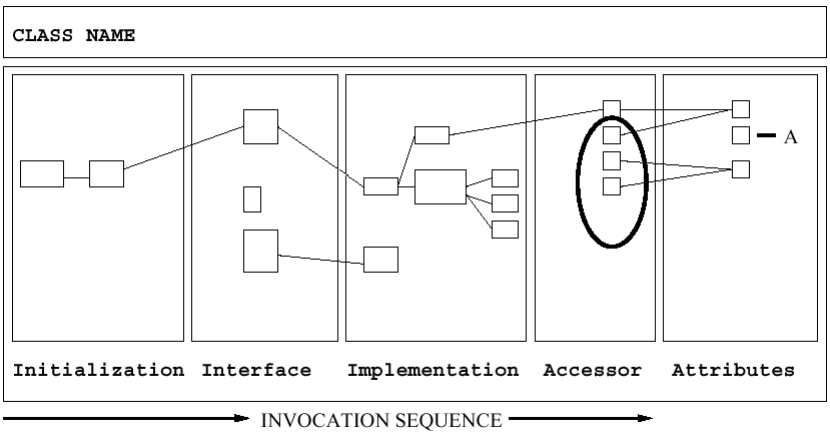


Class Blueprint

- Layers
  - Initialization
    - Methods with substring „init“ or „initialize“
    - constructors
  - Interface
    - Methods invoked by initialization layer
    - „public“ and „protected“ methods
    - Methods not invoked by other methods within the same class
  - Implementation
    - „private“ methods
    - Methods invoked by other methods in the same class
  - Accessor
    - Methods to get and set the values of attributes
  - Attributes
    - All attributes of the class

A Categorization of Classes based on the Visualization of their Internal Structure: the Class Blueprint
Michele Lanza, Stephane Ducasse. Published in the OOPSLA 2001 Proceedings (Conference on Object-Oriented Programming, Systems, Languages, and Applications), pp. 300 - 311, ACM, 2001.

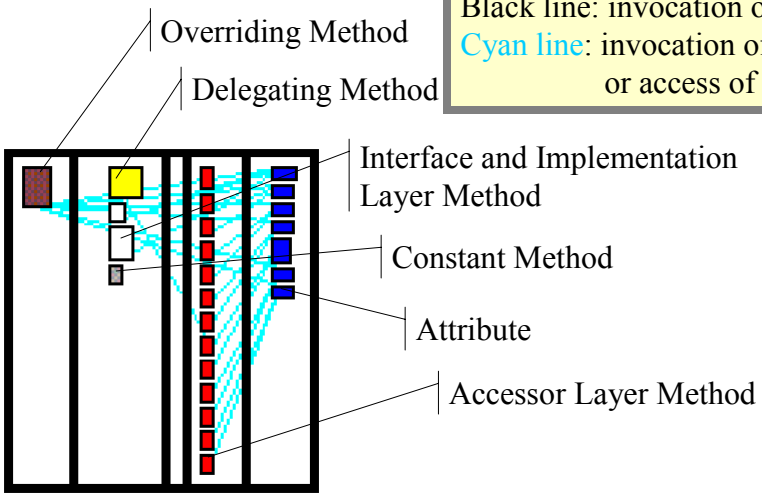
Class Blueprint



Class Blueprint

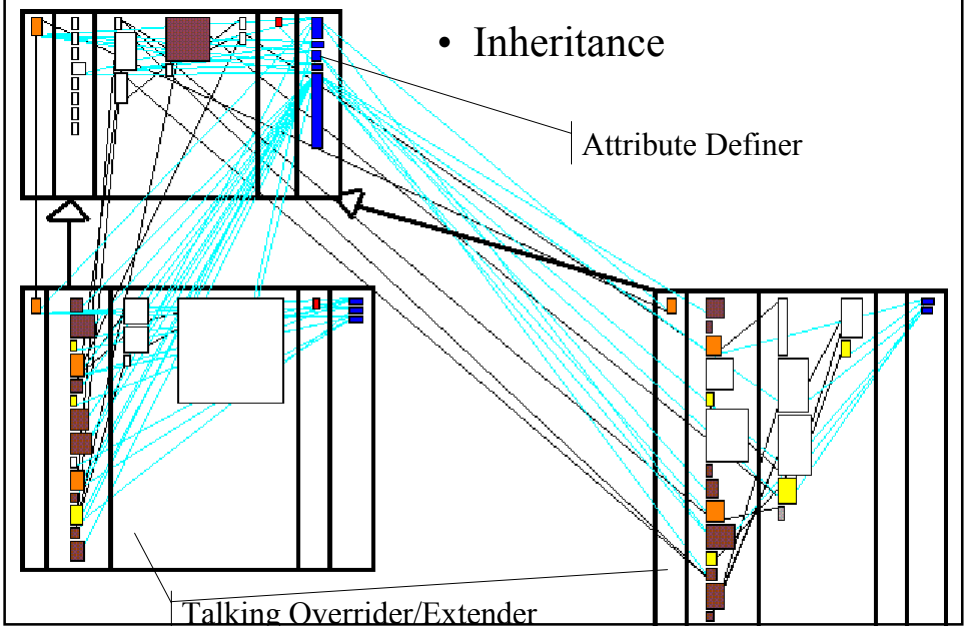
- Colors

Cyan: abstract method
Orange: extending method
Black line: invocation of a method
Cyan line: invocation of an accessor or access of an attribute



Class Blueprint

- Inheritance



Class Blueprint: Categorization

- Based on the blueprint classes can be categorized, e.g. as
  - large implementation,
  - wide interface
  - or delegator classes.
- Taking inheritance into account they can be classified as
  - definers,
  - overriders
  - and extenders.
- Overriders and extenders can be
    - talking (invoking superclass methods)
    - or mute (no invocation of superclass methods).

Exercise

- For the given Java program
  - Draw an architecture diagram using icons and metaphors related to the application. You can draw the diagrams by hand, but you can also produce it with your computer. The best diagrams will be shown in class.
  - Draw the UML class diagram with aggregations.
  - Draw a class blueprint (no color coding required).
- Optional: Search for architecture diagrams on the web and identify familiar architectures therein.

Excerpts of Java Source Code

A pet door that detects animals wearing a collar key (electronically transmitted id). Opens and closes automatically

```
public class Pet
{ int collarKey;  String name; }

public class Pets
{ final int maxPets=10;
  Pet[] list = new Pet[maxPets];
  public boolean contains(int k) { ... }
  public void add(Pet p) { ... }
  public void remove(int k) { ... }
}

public class Door
{ boolean isOpen;
  public void isOpen() { return isOpen; }
  public void open() { isOpen=true; }
  public void close() { isOpen=false; }
}
```

```
public class PetDoor extends Door
{ Pets currentPets, registeredPets;
  PetDoor(Pets regPets)
   { registeredPets=regPets;
     currentPets=new Pets();
     isOpen=false;  }

 public void open()
   { if (!isOpen) { super.open(); } }

 public collarKeySignalReceived(int k)
    { Pet p=registeredPets.contains(k);
      if (p!=null) { open(); currentPets.add(p); }  }

 public collarKeySignalLost(int k)
    { if (currentPets.contains(k))
        { currentPets.remove(k); close(); } }
}
```