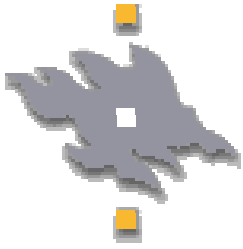


Lecture #13: 15th March 2004

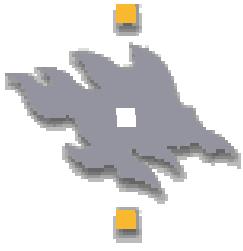
REST

- An Overview -
Suresh Chande



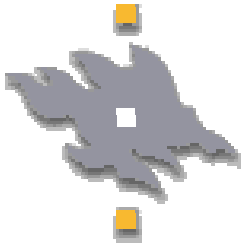
Overview

- What's REST ?
- REST in comparison to Web Services
- Principles
- Guidelines
- Motivation
- Approach
- Architectural Elements
- Architectural properties
- Classification of Architecture
- REST derived from the architectural styles
- Architectural Views



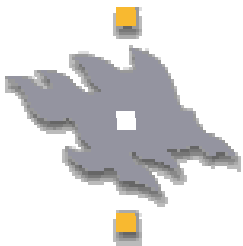
What's REST ?

- **RE**presentational **S**tate **T**ransfer(REST) is a term coined by Roy Thomas Fielding in his PhD Disserttation.
[<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>]
- REST is an architectural style and not a standard, protocol or a solution.
- REST is an architectural style defining the principles of distributed networked systems. REST is the underlying architectural model guiding the design and development of the current and the next generation Web architectures.

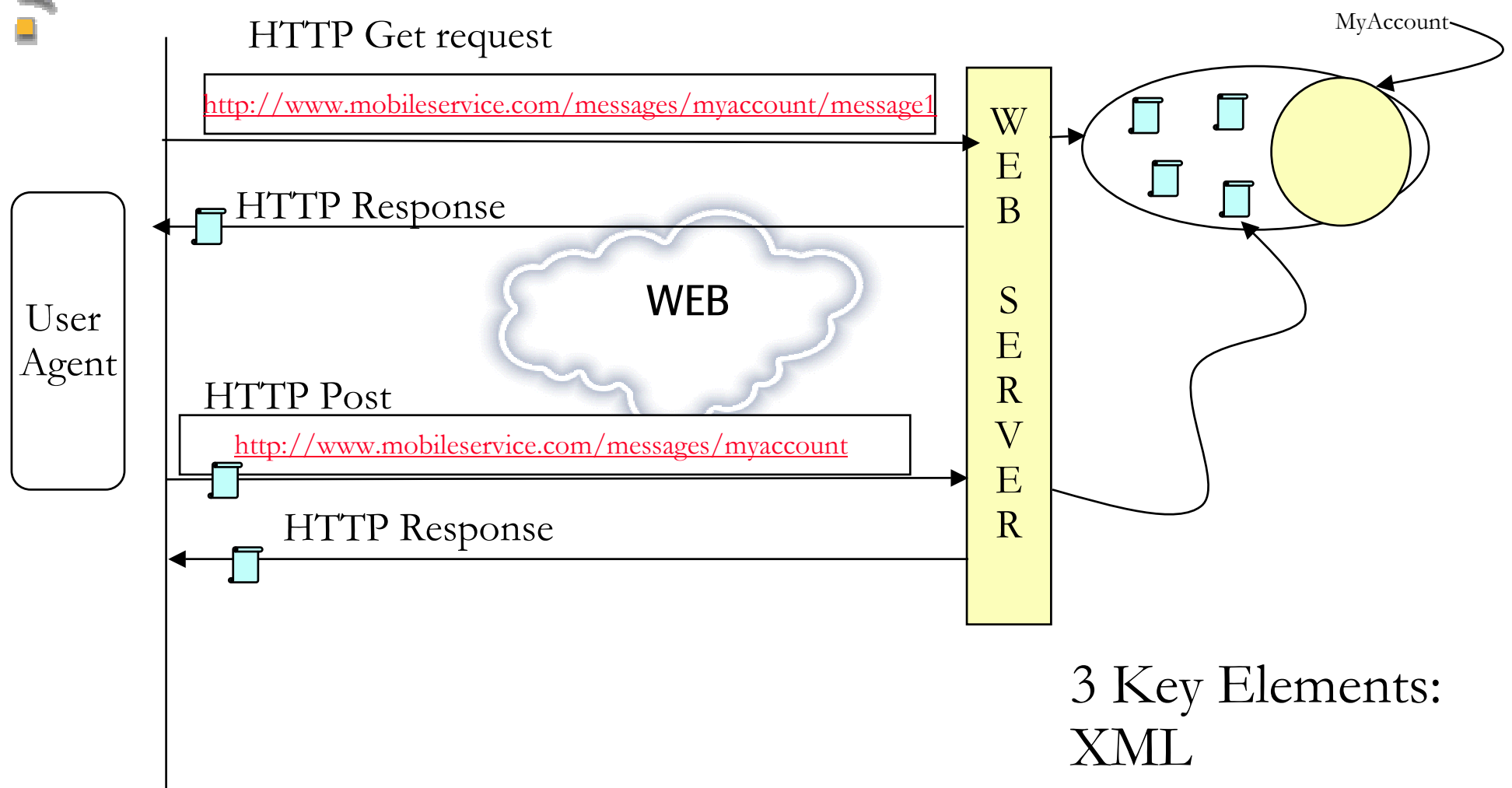


What's REST ?

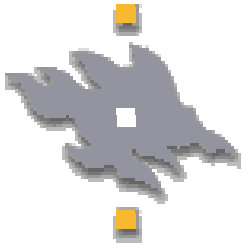
- ”REST provides a set of architectural constraints that, when applied as a whole, emphasizes:
- Scalability of component interactions,
- Generality of interfaces,
- Independent deployment of components,
- Intermediary components to reduce interaction latency,
- Enforce security, and
- encapsulate legacy systems.”



REST by Example

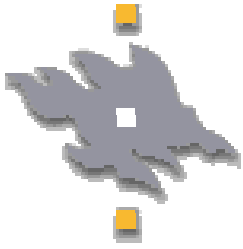


3 Key Elements:
XML
URI
HTTP



REST in comparison to Web Services

- REST promotes and recommends generic operations on resources
 - HTTP methods: PUT, GET, POST, DELETE
 - SQL: select, create, drop, etc
 - Utilises the caching mechanism
 - Standard manipulation of resources based on URI
- WS does not provide generic operations
 - First generation only utilises the HTTP Post, this will change with SOAP1.2(RESTFul)
 - Each services defines it's own application specific operations
 - Requires additional means of description, discovery mechanisms on top of the web
 - Knowledge and semantics of operation is required by the client
 - No Caching capabilities
 - Manipulation of data is based on the message internal structure and application protocols



REST in comparison to Web Services

REST

- REST utilizes URL to identify the desired resource
- Security in REST can be implemented by standard and traditional solutions for authorized access to certain web resources
- In REST every entity in the web is centered around resources
- A better approach for open systems
- Is an Architectural Style
- WRDL provides the description to the web resource

WS

- WS uses SOAP enveloping to identify the desired procedure to be invoked
- Security in SOAP requires additional infrastructure in Web to enable message/transport level security concerns
- In Web Services every entity is centered around interfaces and messages that are channeled into the interface.
- Is a good approach for closed system
- Is an RPC/Document oriented architectures
- WSDL provides the description to Services interfaces which can receive and deliver SOAP Messages



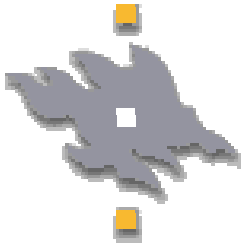
REST (web architectural) Principles

- Web consists of addressable resources. When a user utilizing an applications selects a specific address(URL) a specific **representation** of that resource is returned over the Web. This representation places the client application into a specific **state**. On accessing another URL the client application gets another representation of the resource and in turn **transferring** the state from the current to the new state.



REST (web architectural) Principles

- "Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use." (Roy T Fielding)



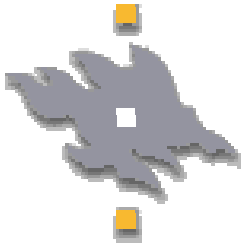
REST principles

- **Stateless Conversation:** All the requests originating from a client and server should contain all the required context for the communication and can not take advantage of any stored data.
- **Cacheable Representations:** Resource representations should be able to be flagged as cacheable/non-cacheable in order to improve the network efficiency.
- **Resource Addressability:** The resources are addressable using URL's and are in turn interconnected via URLS
- **Intermediaries:** There could be several intermediaries introduced between the client and the ultimate resource, such as proxies, gateways, for caching, security, performance, etc
- **Resource Discovery:** Standardized manner of discovering using distributed naming authorities(DNS) and referring to resources (URI)



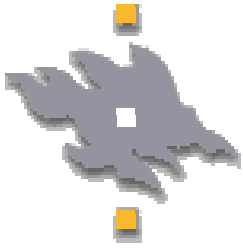
REST Guidelines

- View any web based system as a set of **resources addressable as URLs**
- Use **URIs to refer to specific resources** (refer to resources as **nouns and not as verbs**)
 - Noun example: http://www.mobile.services/ringtone/LooneyTune_v1_0
 - Verb Example: http://www.mobile.services/getRingTone?tune=LooneyTunev1_0 X
- **Prefer an URI which are logical instead of physical**
 - Logical: http://www.mobile.services/ringtone/LooneyTunev1_0
 - Physical: http://www.mobile.services/ringtone/LooneyTunev1_0.html
- **Minimize the use of Query Strings**
- Utilize the appropriate **generic operations** to perform the resource manipulation
 - GET for simple read operations
 - POST for modifying the state of the resource



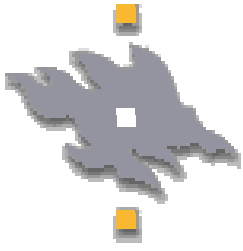
REST Guidelines

- **Represent resources** using **schemas** with inbuilt capability for extension included.
- Servers and Intermediaries should be **stateless**
- Clients need to maintain the state and state transitions



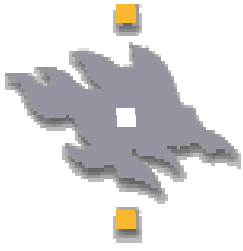
Motivation

- Software Design and Architectural principles guides software community on:
 - Separation of concerns,
 - Modelling the software systems,
 - Definition & Maintenance of system communications
 - Evolution of the systems independently
 - Scalability



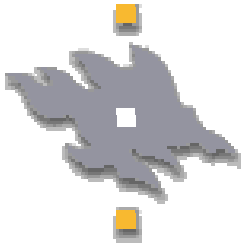
Motivation [contd..]

- Web being a Distributed networked infrastructures,
- How can we through principled constraints over architectural styles ensure and provide architectural guidance to the web community to maintain:
 - Scalability of component interactions
 - Generality of interfaces
 - Independent deployment of components
 - Intermediary components to reduce interaction latency,
 - Enforce security &
 - Encapsulate legacy systems.



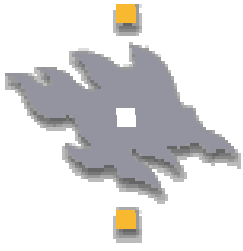
Approach

- Fielding defines firstly a framework for understanding the software architectures by means of defining architectural styles, the styles that can be used to guide the architectural design of the network-based applications.
- Architectural styles are classified into the architectural properties they will induce when applied into architecture for distributed systems(hypermedia)
- REST is one such architectural style which has guided the design of the modern Web Architectures. The Styles are also to guide the future expansions and further develop of that particular application



Architecture

- A software architecture is an abstraction of the run-time elements of a software system. The system can contain several levels of such abstraction with each level represented by its own sub architecture.
- An architecture specifies software entities which are meant to process(component) and connect (connectors) data across the different architectural components.
- An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.



Architectural properties

- **Purpose:** To differentiate and classify Architectural styles
- **Properties:**
 - **Performance:** This is dictated by the Application requirements, Interaction style, realized architecture & implementation.
 - Network Performance
 - User-percieved Performance
 - Network Efficiency
 - **Scalability:** Solved by distribution of centralized interactions
 - Affect Factors: frequency of interaction, Load, Reliability, sychronous or asynchronous, Clustering
 - **Simplicity**
 - **Modifiability**
 - Evolveability, Extensibility, Customizability, configurability, reusability
 - **Visibility**
 - **Portability**
 - **Reliability**



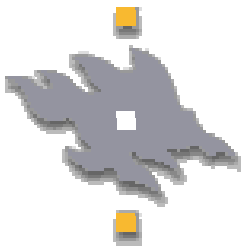
Classification of the Architectural Styles

- Data-Flow Styles
 - In the Pipe & Filter the system operates on a Stream of data through one or more independent software components (filters), each of which applies some transformation to the data. Uniform pipe and filter (UPF) restricts that any filter to follow a uniform interface,, much as systems in UNIX with standard data streams stdin,stdout&stderr
- Replication Styles
 - Replicated Repository(RR) Style improves accessibility to data and service response time by making the same data to be provided by more than 1 process; for example: distribute File System, CVS, Clustering
 - Caching

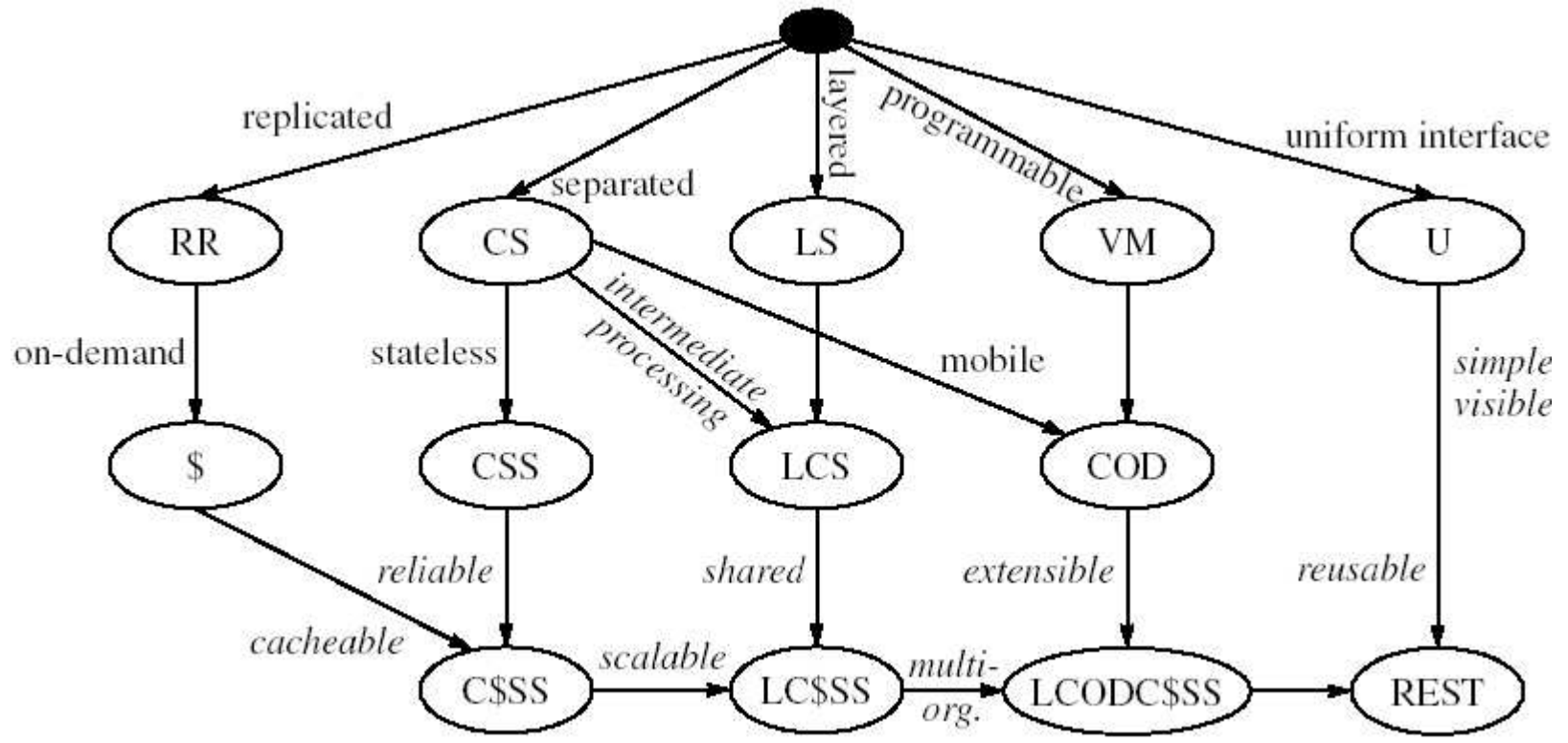


Classification of the Architectural Styles

- Hierarchical Styles
 - Client-Server (CS)
 - Layered System/Client-Server (LS)
 - Client-Stateless Server:
 - Client Cache-Stateless Server: NFS
 - Layered Client Cache Stateless Server (LCS): DNS
 - Remote Session: Telnet, FTP
 - Remote Data Access: database systems: SQL
- Mobile Code Styles
 - Virtual machine (VM)
 - Remote Evaluation
 - Code on Demand (CoD)
 - Layered CoD Client Cache Stateless Server (LCODC\$SS)
 - Mobile Agent
- Peer-to-Peer Styles
 - Event Based Integration and C2
 - Distributed Objects(DO) & Brokered DO



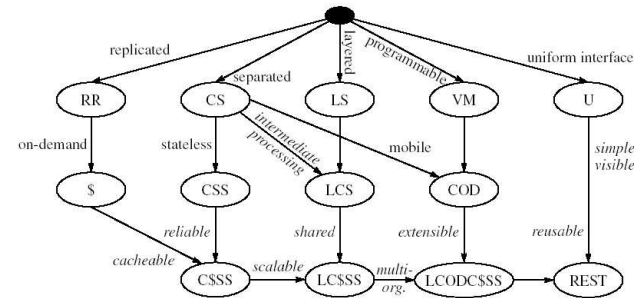
REST derived from the Architectural Styles



Source: Thomas Roy Fielding's PhD dissertation

REST derived from the Architectural Styles

- Based on Client Server
 - Separation of Concerns
- Stateless
 - Visibility, Reliability and Scalability
- Cache
- Uniform Interface between components
- Layered systems allowing the possible to include intermediaries, such as: proxies, gateways and shared caches
- Code on Demand



Source: Thomas Roy Fielding's PhD dissertation



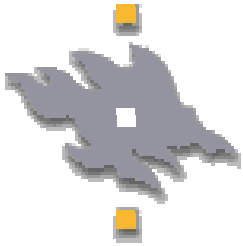
REST Architectural Elements

- REST is an abstraction of architectural elements which forms the basis for components to connect to transfer representation of resources as basic data elements
- The three architectural elements:
 - **Data Elements**
 - **Connectors**
 - **Components**
- **Data elements**
 - **Resource:** Conceptually the Target of the hypertext reference
 - **Resource identifier:** URI, URL, URN
 - **Representation:** HTML, JPEG
 - **Representation Metadata:** Content-Type, Content-Size, etc
 - **Resource Metadata:** Alternatives
 - **Control Data:** Cache control, etc



REST Architectural Elements

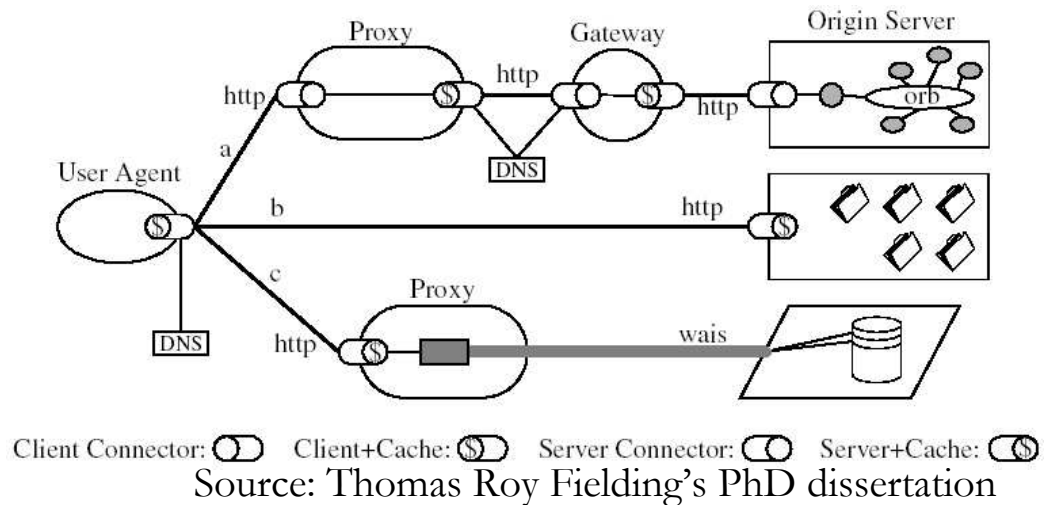
- **Connectors:** Various types to encapsulate the activities of accessing and transferring resources
 - Client
 - Server
 - Cache
 - Resolver
 - Tunnel
- **Components**
 - **Origin Servers:** Source of representations of its resources and an ultimate recipient of request which modifies/alters the value of resource.
 - **Gateway:** Intermediary imposed by the network or origin server for Interface encapsulation, data transformation, performance enhancements, security enhancements.
 - **Proxy:** Intemediary selected by the client for: Interface encapsulation, data transformation, performance enhancements, security enhancements.
 - **UserAgent:** A Client connector to orginate the request and the recipient of the response



REST Architectural Views

Combination of the elements together to Architectures

3 Different Views :



- **Process View:**

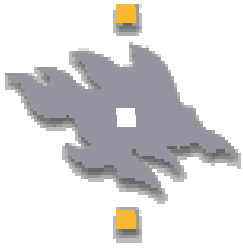
- Shows the interactions amongst the various components of the architecture and in turn reveals the flow of data among the different architectural elements. Interaction of a real system usually involves an extensive number of components, resulting in an overall view that is obscured by the details.

- **Connector View:**

- Highlights the mechanics of the communication between components.

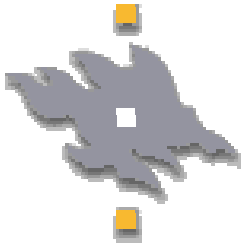
- **Data:**

- Reveals the application state as information flows through the components.



Web Application Requirements

- Web : A Shared information space where people and machines could communicate
- Basic Requirements :
 - Low Entry Barrier
 - Extensibility
 - Distributed Hypermedia
 - Scalability of Internet scale
- The Web is an ideal application of the REST Architecture



Further Reading

- Architectural Styles and the Design of Network-based Software Architectures, PhD Dissertation, Thomas Roy Fielding, 2000, URL: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- REST WIKI : A dedicated site to discuss Rest Architectural Styles: <http://internet.conveyor.com/RESTwiki/moin.cgi/FrontPage>
- A good tutorial by Roger L. Costello : <http://www.xfront.com/REST.ppt>
- REST site by Paul Prescod : <http://www.prescod.net/rest/>
- SOAP vs REST: http://www.prescod.net/rest/rest_vs_soap_overview/