Definition: **Mashup** simply indicates a way to create new Web applications by combining existing Web resources, utilizing data and Web APIs.

**Mashup Techniques**

- Work for the combination of data and services can be done on the server, the client, or both of them.
  1. **Mashing on the Web Server**
  2. **Mashing using Ajax**
  3. **Mashing with JSON**

**Development Support**

Yahoo Pipes are hosted and executed on a Yahoo server.

GeoWiki was a Wiki-based mashup maker by IBM, pages are hosted on an IBM server, mostly executed on the client side.

ProgrammableWeb provides a mashup directory and marketplace which let users rank and discuss mashups.

Web Services

Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the W3C community.

Motivation - Example

Service Oriented Architecture - SOA
Web Services - Characteristics
Web Services Architecture
Simple Object Access Protocol (SOAP)
Web Services Description Language (WSDL)
Universal Description, Discovery, and Integration (UDDI)
REST

Web Service Composition
Adopting Web Services
Mashups

**1 Mashing on the Web Server**

All the work of mashing is done on a Web server while the browser just waits for a response.

**Characteristics**

- Browser is decoupled from the partner sites supplying the data.
- Web server acts as a proxy and aggregator for the responses.
- Browser requests the entire page.
- Scalability problem because server does all the work.
This approach allows a richer user experience; the work is divided between the server and the browser.

**Characteristics**
- more complex because developers face JavaScript challenges, server communication and asynchronicity.
- Ajax may refresh only a portion of the page.
- navigation mechanism of browser is bypassed.
- approach may result in a rich Internet application.

**Mashing with JSON**

**JSON** (JavaScript Object Notation): lightweight data-interchange format that is gaining popularity in the mashup community.

**Mashups**

**Definition**: Mashup simply indicates a way to create new Web applications by combining existing Web resources utilizing data and Web APIs.

**Mashup Techniques**
- Work for the combination of data and services can be done on the server, the client or both of them.

1. **Mashing on the Web Server**
2. **Mashing using Ajax**
3. **Mashing with JSON**

**Development Support**
Yahoo Pages are hosted and executed on a Yahoo server.

*Wiki* was a Wiki-based mashup maker by IBM, pages are hosted on an IBM server, mostly executed on the client side.

*ProgrammableWeb* provides a mashup directory and marketplace which let users rank and discuss mashups.
Development Support

In order to facilitate and speed up mashup development, a number of tools and frameworks have recently emerged. Two dimensions may be distinguished:

**Component Model**: describes the characteristic properties of the mashup components:
- type: a component can be data, application logic, or user interface.
- interface: create-read-update-delete (CRUD) interface, API for a specific programming language or IDL/WSDL.
- extensibility: whether the user may extend the component model.

**Composition Model**: specifies how the components are glued together to create the mashup application:
- flow-based: defines the orchestration as sequencing or partial order among components.
- event-based: uses the publish-subscribe model.

Examples for tool-assisted mashup development:

- **Yahoo Piped**: mix data feeds to create data mashups using a visual editor.

Mashup Techniques

- **Mashing on the Web Server**
- **Mashing using Ajax**
- **Mashing with JSON**

Deblup Support

Yahoo Pipes are hosted and executed on a Yahoo server.

GeoWiki was a wiki-based mashup maker by IBM; pages are hosted on an IBM server; mostly executed on the client side.

Programmable Web provides a mashup directory and marketplace which let users rank and discuss mashups.

---

Distributed Applications - Verteilte Anwendungen

- **Prof. J. Schlichter**
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
  - Boltzmannstr. 3, 85748 Garching
  - Email: schlichter@in.tum.de
  - Tel.: 089-289 18654
  - URL: http://www11.in.tum.de/

Overview

**Introduction**

**Architecture of distributed systems**

**Remote Invocation (RPC/RMI)**

**Basic mechanisms for distributed applications**

**Web Services**

**Design of distributed applications**

**Distributed file service**

**Distributed Shared Memory**

**Object-based Distributed Systems**

**Summary**

Issues

1. Assignment of names to addresses.
   - What should happen if a client cannot contact the localized server subsystem?

2. Communication mechanisms
   - Selection of the desired communication model, e.g., client-server model, group communication or peer-to-peer.
   - How does the application (both client and server) handle network communication errors?

3. Consistency
   - How can the data be kept consistent, particularly for replicated data?
   - If a cache is used for performance improvement, then it must be kept consistent with the stored data.
   - User interface consistency for the individual components.

4. User requirements
   - Functionality and reconfigurability of the distributed application and its components.
   - Service quality, such as security, reliability, fault tolerance and performance.
   - What kind of security mechanisms are provided? Is authentication an issue?
   - Which actions will be triggered if a client cannot communicate with its server?
   - What type of heterogeneity is necessary?
   - What efficiency requirements are present?
Designing a distributed application is a 7-step approach:

1. The repositories of the application data are identified.
2. Data are assigned to individual modules. This is a fundamental step of any software engineering approach.
3. The module interface is defined.
4. Define a network interface.
5. Classify each module as client or server.
6. Registration of servers, i.e. the method in which servers are to be made available to other functional units is determined.
7. A strategy for the binding process of client and server subsystems is defined.