http://ks.fernuni-hagen.de/

Szenarien und Beispiele moderner Integrationslösungen

Jan Gellweiler

03.12.2007

CampusSource-Workshop in Dortmund

Agenda

- **Begriffsbestimmung Integration / Kopplung**
- **Integrationslösungen und Beispiele**
- **Demonstration von Mashups anhand eines grafischen Editors (www.popfly.ms)**

Was ist Integration?

- **Der Begriff Integration wird in vielen Zusammenhängen verwendet.**
- **Hier findet die Definition aus der Informatik / Softwaretechnik Anwendung.**
- **Die Integration dient der Verknüpfung von verschiedenen Anwendungen.**
- **Ziel der Integration ist die Verringerung und Vermeidung von Schnittstellen.**
- **Man unterscheidet die folgenden Varianten der Integration:**
  - Datenintegration
  - Funktionsintegration
  - Geschäftsprozessintegration

Datenintegration

Bei der Datenintegration greifen alle beteiligten Applikationen auf ein einheitliches Datenmodell zu.

Vorteile:

- **Einheitliche Semantik für alle beteiligten Applikationen.**
- **Redundanzen werden vermieden.**

Nachteile:

- **Es entstehen große, evtl. schwer wartbare Systeme.**

Funktionsintegration

Bei der Funktionsintegration werden mehrere Applikationen, die zur Erfüllung einer gemeinsamen Aufgabe eingesetzt werden, in einer Anwendung zusammengefasst.

Vorteile:

- **Schnittstellen zur Datenübergabe werden vermieden.**
- **Funktionen müssen nicht mehrfach implementiert werden.**

Nachteile:

- **Es entstehen große, evtl. schwer wartbare Systeme.**

Geschäftsprozessintegration

Bei der Geschäftsprozessintegration werden die benötigten IT-Komponenten entlang modellierter Geschäftsprozesse mit Hilfe einer Integrationsplattform (Middleware) verknüpft.

Vorteile:

- **Applikationen bleiben in sich unverändert.**
- **Integration von Standartsoftware ist möglich.**

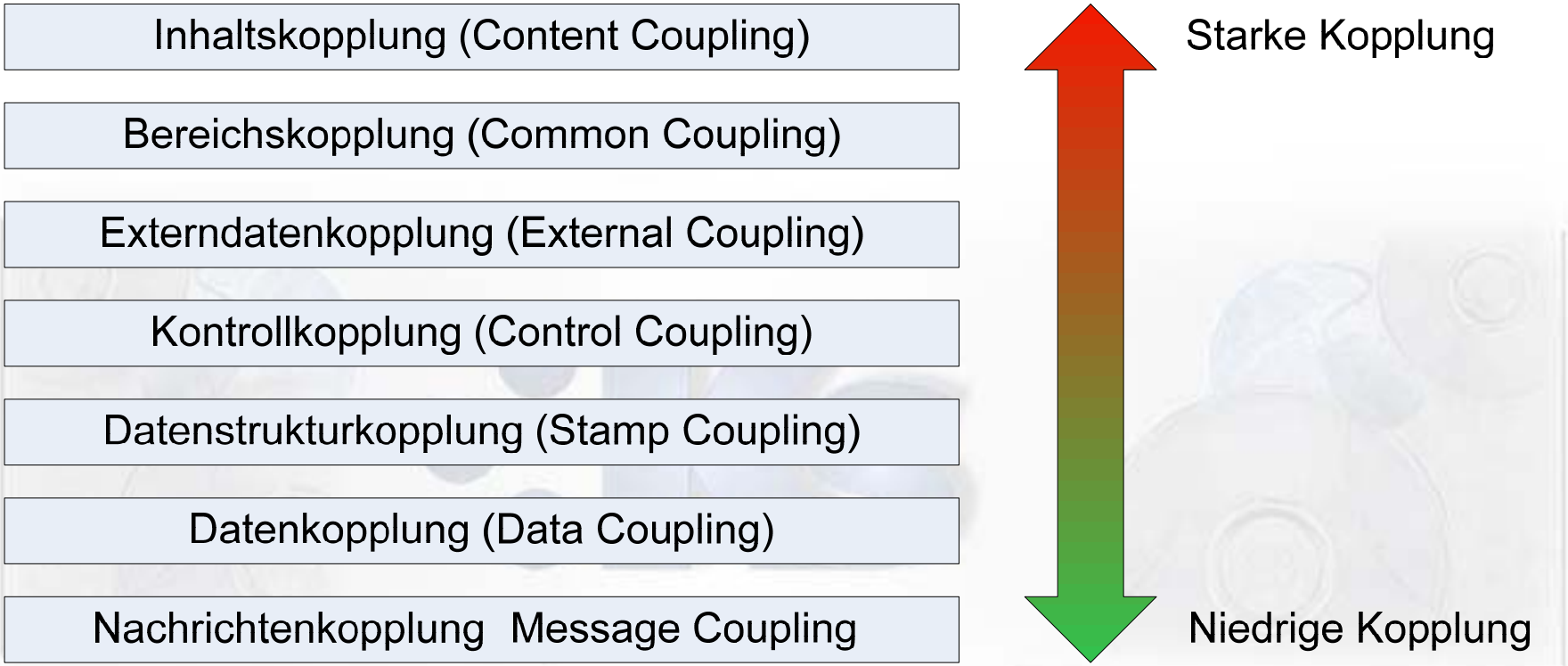
Nachteile:

- **Notwendige Adapter zur Abstraktion von Schnittstellen, zum Teil aufwendig zu implementieren.**
- **Geschäftsprozesse müssen formal erfasst werden / sein.**

Was ist Kopplung?

- In der Informatik die Verknüpfung von Anwendungen unter Verwendung von Schnittstellen.
- Schnittstellen werden nicht abgebaut (vgl. Integration) sondern verknüpft / gekoppelt.
- Hauptproblem der Kopplung ist die Form der Datenübergabe.
- Die Formen der Datenübergabe können sehr unterschiedlich sein und reichen vom menschlichen Eingriff über einfache Dateien (z.B. im XML-Format) und Konverter bis zu Kopplungsprozeduren (z.B. mittels RPC, CORBA oder Web Services).
- In der objektorientierten Programmierung beschreibt die Kopplung den Grad der Abhängigkeit von Klassen. Ziel ist eine möglichst geringe Abhängigkeit (lose Kopplung).

Arten der Kopplung



Je niedriger die Kopplung, desto eigenständiger die Anwendung, die Klasse, das Modul, usw.

Enge (Starke) Kopplung

Vorteile:

- Ständige Verfügbarkeit aller Komponenten kann vorausgesetzt werden.
- Vereinbarungen über Aufrufsyntax und Datenformate ändern sich nicht überraschend.
- Optimale Performance durch exakte Abstimmung der Komponenten aufeinander möglich.

Nachteile:

- Jede Änderung an einer Komponente muss in allen gekoppelten Komponenten nachvollzogen werden.
- Ständige Verfügbarkeit für verteile (evtl. über das Internet verbundene) Anwendungen kaum zu gewährleisten.
- Eine Trennung eng gekoppelter Systeme ist kaum möglich.

Lose (Niedrige) Kopplung

Vorteile:

- Es brauchen nur wenige Annahmen / Vorrausetzungen gemacht werden. (Bzw. es sollen möglichst wenige Annahmen gemacht werden.)
- Komponenten agieren (möglichst) autonom und können leicht wieder getrennt werden.
- Die (interne) Implementierung der weiteren Komponenten muss nicht bekannt sein.
- Lose gekoppelte Systeme sind prinzipiell leichter und schneller zu entwickeln.

Nachteile:

- Daten sind aufgrund der losen Kopplung evtl. nicht immer über alle beteiligten Systeme hinweg konsistent.
- Zeitlich lose Kopplung erschwert die Transaktionskontrolle.

Fazit Integration vs. Kopplung

- Der Begriff Integrationslösungen umfasst beide Konzepte.
- Die Grenze zwischen Integration und Kopplung ist im Bereich der Geschäftsprozessintegration fließend.
- Das Konzept der losen Kopplung (entlang der Geschäftsprozesse) ist das vorherrschende Prinzip moderner Integrationslösungen.

Agenda

- Begriffsbestimmung Integration / Kopplung
- **Integrationslösungen und Beispiele**
- Demonstration von Mashups anhand eines grafischen Editors (www.popfly.ms)

Application to Application Integration (A2A)

- Anwendungen werden paarweise integriert oder über Schnittstellen gekoppelt.
- Jede neue Anwendung benötigt Schnittstellen zu jeder bestehenden Anwednung.
- Wird bei einer Anwendung die Schnittstelle modifiziert, müssen alle verbundenen Schnittstellen ebenfalls angepasst werden.
- Keine moderne Integrationslösung!

Enterprise Application Integration (Konzept)

- Konzept zur unternehmensweiten Integration der Geschäftsfunktionen entlang der Wertschöpfungskette.
- Strikte Trennung von Geschäftsprozesslogik und Geschäftsfunktionen.
- Umfasst Planung, Methoden und Software.
- In der klassischen Form eine IT-Architekturlösung.
- Ausgerichtet auf die Integration bestehender Fachanwendungen / Investitionsschutz.

Enterprise Application Integration (Integrationsmodelle)

- **Datenintegration (Data Level Integration)**
  - Integration von Systemen über den Austausch von Informationen auf der Ebene der Datenhaltung.
  - Anpassung bzw. Erweiterung der einzelnen Systeme meist nicht notwendig.
  - Integration erfolgt manuell in der Zielanwendung oder durch definierte Systemschnittstellen.
  - Enge Kopplung.
- **Anwendungsintegration (Message Level Integration)**
  - Integration auf der Grundlage eines Nachrichtenaustauschs.
  - Asynchronen Kommunikation / Lose Kopplung der Systeme.
  - Schnittstellen zum Senden und Empfangen von Nachrichten notwendig.
  - Reduziert die Entwicklungskosten insb. die Wartung der Integrationen stark.
  - Leichte Erweiterbarkeit bzw. Anpassbarkeit der Systeme gewährleistet.
- **Prozessintegration (Process Level Integration)**
  - Unterstützt die Integration auf Basis von Geschäftsprozessen.
  - Die EAI Middleware (Business Bus) agiert dabei als eine Prozess- oder Workflow-Engine mit flexibel definierten und dynamisch interpretierten Geschäftsprozessen.
  - Flexible Anpassung an aktuelle Geschäftsmodelle möglich.

Enterprise Application Integration

Vorteile:

- Zentral kontrollier- und wartbare Integrationsdrehscheibe.
- Erbt bei der Anwendungs- und Prozessintegration die Vorteile der losen Kopplung.
- Paradigma der Prozessorientierung.
- Prozesse sind unabhängig von Applikationen.

Nachteile:

- Hohe Anfangsinvestitionen und lange Implementierungszeiten.
- Strategisch und operativ eine große Herausforderung.
- In der klassischen Form eine Infrastruktur.
- Mangelnde Standardisierung.

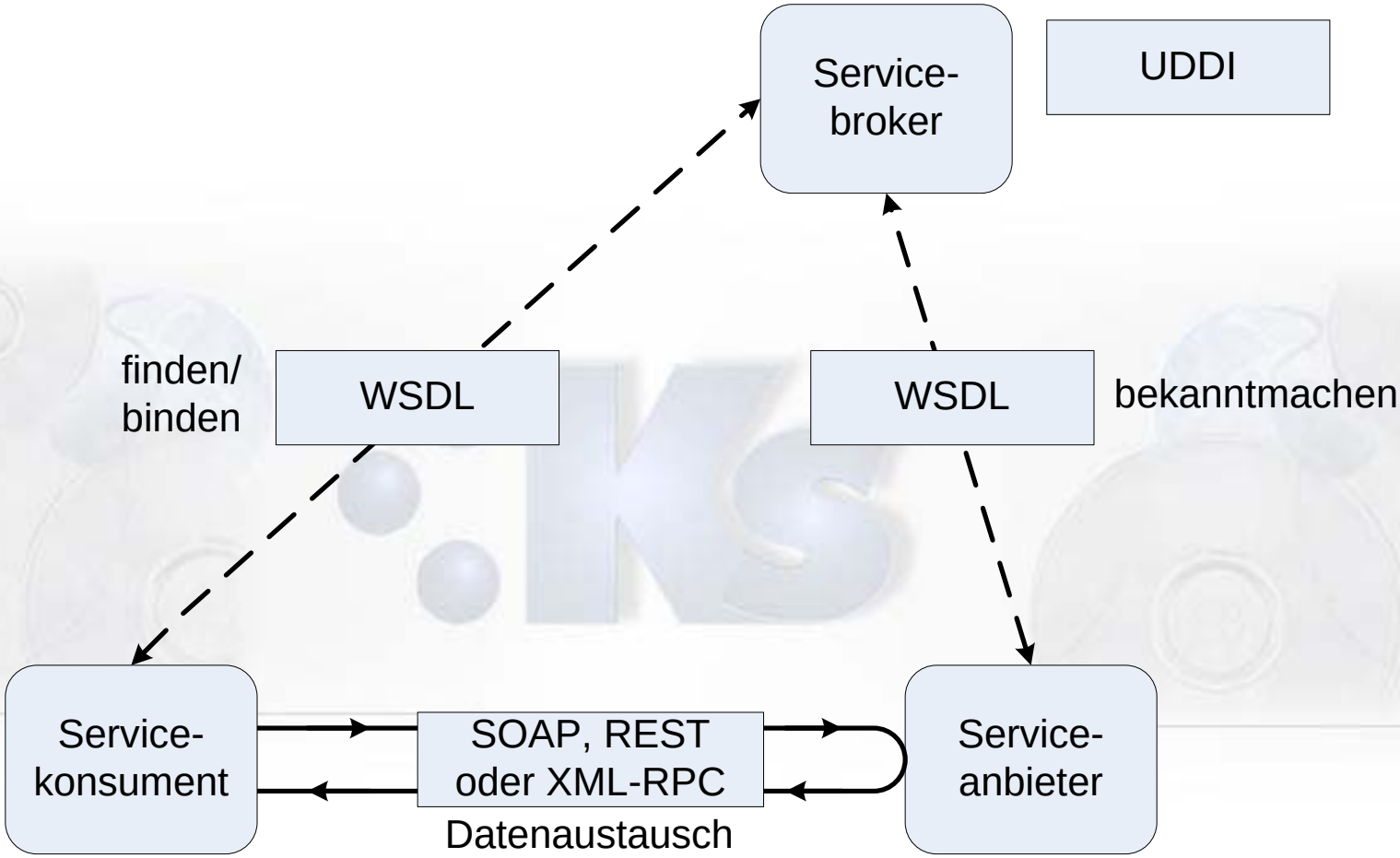
Web Services (1)

Ein Web Service ist eine Komponente, die ihre Funktionalität über eine veröffentlichte Schnittstelle anbietet und über ein offenes, im Internet verwendetes Protokoll zugreifbar ist.*

- **Paradigmenwechsel von Anwendungen, Funktionen und Methoden hin zu Diensten.**
- **Direkte Interaktion mit anderen Services unter Verwendung von XML-basierten Nachrichten (lose Kopplung).**
- **Selbstbeschreibend durch Metadaten.**
- **Kann in Verzeichnissen / Repositories registriert werden.**
- **Realisierung mit SOAP, XML-RPC oder REST.**
- **Rollen: Konsument, Anbieter, Broker/Verzeichnis**
- **Einsatzbereich ist die Kommunikation zwischen Anwendungen, nicht die Kommunikation zwischen Anwendung und Anwender.**

* Nach http://www.jeckle.de/webServices/index.html#def

Web Services (2)



UDDI: Universal Description, Discovery and Integration
WSDL: Web Service Description Language
SOAP: Simple Object Access Protocol

Web Services (3)

Vorteile:

- Verwendung offener Standards
- Offene, flexible Architektur
- Vergleichsweise niedrige Einstiegshürde

Nachteile:

- Mangelnde Performance (XML parsen, Overhead)
- Mangelnde Sicherheit
- Mangelnde Transaktionsfähigkeit

Web Services / SOAP (1)

- **Simple Object Access Protocol**
- **Übertragung des Remote Procedure Call (RPC) Gedanken ins Web**
- **Austausch von Daten / Durchführung von RPC**
- **RPC Middleware**
- **Setzt auf Standards auf:**
  - HTML oder SMTP als Transportprotokoll
  - XML als Nachrichtenformat
- **Alle Nachrichten werden über eine zentrale Instanz (SOAP Dispatcher) geroutet.**

Web Services / SOAP (2)

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2001/12/soap-envelope">
    <s:Body>
        <m:TitleInDatabase xmlns:m="http://www.seminar-db.de/soap">
            DOM, SAX und SOAP
        </m:TitleInDatabase>
    </s:Body>
</s:Envelope>
```

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2001/12/soap-envelope">
    <s:Header>
        <m:RequestID xmlns:m="http://www.lecture-db.de/soap">a3f5c109b</m:RequestID>
    </s:Header>
    <s:Body>
        <m:DbResponse xmlns:m="http://www.lecture-db.de/soap">
            <m:title value="DOM, SAX und SOAP">
                <m:Choice value="1">Arbeitsbericht Informatik</m:Choice>
                <m:Choice value="2">Seminar XML und Datenbanken</m:Choice>
            </m:title>
        </m:DbResponse>
    </s:Body>
</s:Envelope>
```

Quelle: http://de.wikipedia.org/wiki/SOAP

Web Services / SOAP (3)

Vorteile:

- Leichtgewichtiges Protokoll
- Flexibel einsetzbar
- Nicht Architekturabhängig
- Nicht auf Web Services beschränkt
- Status einer W3C-Empfehlung

Nachteile:

- Übertragungsvolumen
- Rechenaufwand
- Keine direkte Adressierung

Web Services / REST (1)

- Representational State Transfer Protocol
- Vereinfachung der Integration durch Rückbesinnung auf grundlegende Web-Technologien.
- Grundlegende HTTP-Befehle (GET, POST, PUT, DELETE,…) dienen als zentrales Steuerungselement der Integration.
- Anbieter (Server) verfolgen keinen Clientstatus.
- Kommunikation erfolgt nur auf Abruf.
- Manipulation/Abruf von Resourcen (Seiten, Bilder, Servlets …) nur über URI (Universal Resource Identifier) möglich.
- Wird von vielen Suchmaschinen, Shops, etc. eingesetzt. (z.B. Amazon Simple Storage Service, eBay Shopping Web Services)

Web Services / REST (2)

Aufruf

```
GET /warenkorb/5873
```

Repräsentation eines Warenkorbs

```
HTTP/1.1 200 OK
Content-Type: text/xml
<?xml version="1.0"?>
<warenkorb xmlns:xlink="http://www.w3.org/1999/xlink">
  <kunde xlink:href="http://shop.oio.de/kunde/5873">5873</kunde>
  <position nr="1" menge="5">
    <artikel xlink:href="http://shop.oio.de/artikel/4501" nr="4501">
      <beschreibung>Dauerlutscher</beschreibung>
    </artikel>
  </position>
  <position nr="2" menge="2">
    <artikel xlink:href="http://shop.oio.de/artikel/5860" nr="5860">
      <beschreibung>Earl Grey Tea</beschreibung>
    </artikel>
  </position>
</warenkorb>
```

Quelle: http://www.oio.de/public/xml/rest-webservices.htm

Web Services / REST (3)

Vorteile:

- Skalierbarkeit (Bestes Beispiel das WWW)
- Leichte Anbindung von Fremdsystemen
- Unabhängig installierbare Komponenten
- Leichte Komposition von Diensten
- Caching ist möglich
- Volle Kontrolle durch Proxy-Server möglich

Nachteile:

- Performance
- Serialisierung und Deserialisierung von XML ist dem Programmierer überlassen.

Serviceorientierte Architektur

Vier Integrationsschichten einer SOA:

- **Datenintegration (inkl. Meta Daten Integration)**
- **Anwendungsintegration (EAI oder Enterprise Service Bus)**
- **Business Process Management (BPM)**
- **Prozess Management Portale (Mensch-Maschine-Schnittstelle zu den Geschäftsprozessen, zur Präsentation und zur Kollaboration)**

Serviceorientierte Architektur

Vorteile:

- **Flexibilität und Automatisierung bei der Umsetzung von Geschäftsprozessen.**
- **Verwendung erprobter Muster (business pattern).**
- **Kapselung der Anwendungssysteme als wiederverwendbare Bausteine (nicht neu).**
- **Umsetzung auf Basis schlanker, standardisierter Technologien.**
- **Möglichkeit zur evolutionären Migration.**

Nachteile:

- **Nicht vollständig standardisiert.**
- **Erbt bei der Verwendung von Web Services deren Schwächen (Performance, Sicherheit, Transaktionsfähigkeit).**
- **Methodik zur Einführung einer SOA ist noch nicht etabliert.**

Beispiel: Sopera (1)

- **SOA Framework der Deutschen Post.**
- **ToolSuite**
  - Eclipse Erweiterungen für Service-, Prozess- und Policy-Editoren sowie das Deployment
- **Runtime Framework**
  - Verteilter Servicebus (JORAM JMS) mit Registrierungs-, Sicherheits-, Nachrichtentransport- sowie Prozessmanagement Diensten.
  - Schnittstellen zu Geschäftsanwendungen, -prozessen und –daten
- **Management**
  - Service Monitoring und Resource Management
- **Als Open Source an die Eclipse Foundation übergeben.**
- **Weiterentwicklung bei Eclipse unter der Bezeichnung „Swordfish“.**

Portale (1)

„Das ideale Portal eröffnet einen gemeinsamen, personalisierten Zugang zu Daten, Expertisen und Anwendungen“ *

- **Funktionen eines Portals:**
  - Erstellung von Frontends (z.B. mit Portlets)
  - Integration von Anwendungen, Prozessen und Diensten
  - Personalisierung
  - Suchfunktionen
  - Single-Sign-On
  - Dokumentenmanagement
  - (Enterprise-)Content-Management
  - Business Process Management

*Gartner Dataquest

Portale (2)

Vorteile:

- Einheitliche Benutzeroberfläche
- Gemeinsame Datenbasis
- Transparentere und effizientere Prozesse
- Single-Sign-On
- Hohe Aktualität der Daten

Nachteile:

- Integration bestehender System meist nur in Form der reinen Daten
- Fehlende Standardisierung

Mashups (1)

- Engl.: „to mash“ für vermischen, vermaschen
- Technologie auf Basis von Web 2.0
- Inegration von zwei oder mehr Web-Dienstleistungen in einer neuen Anwendung (Weblikation)
- Integration auf Daten-, Logik- oder Darstellungsebene
- Einfache Overlays und Kombinationen
- Kaum zeitliche oder logische Abhängigkeiten
- Durch Verzicht auf komplizierte Orchestrierung eine Art „SOA-Light“
- Auch als Gadgets einsetzbar

Mashups (2)

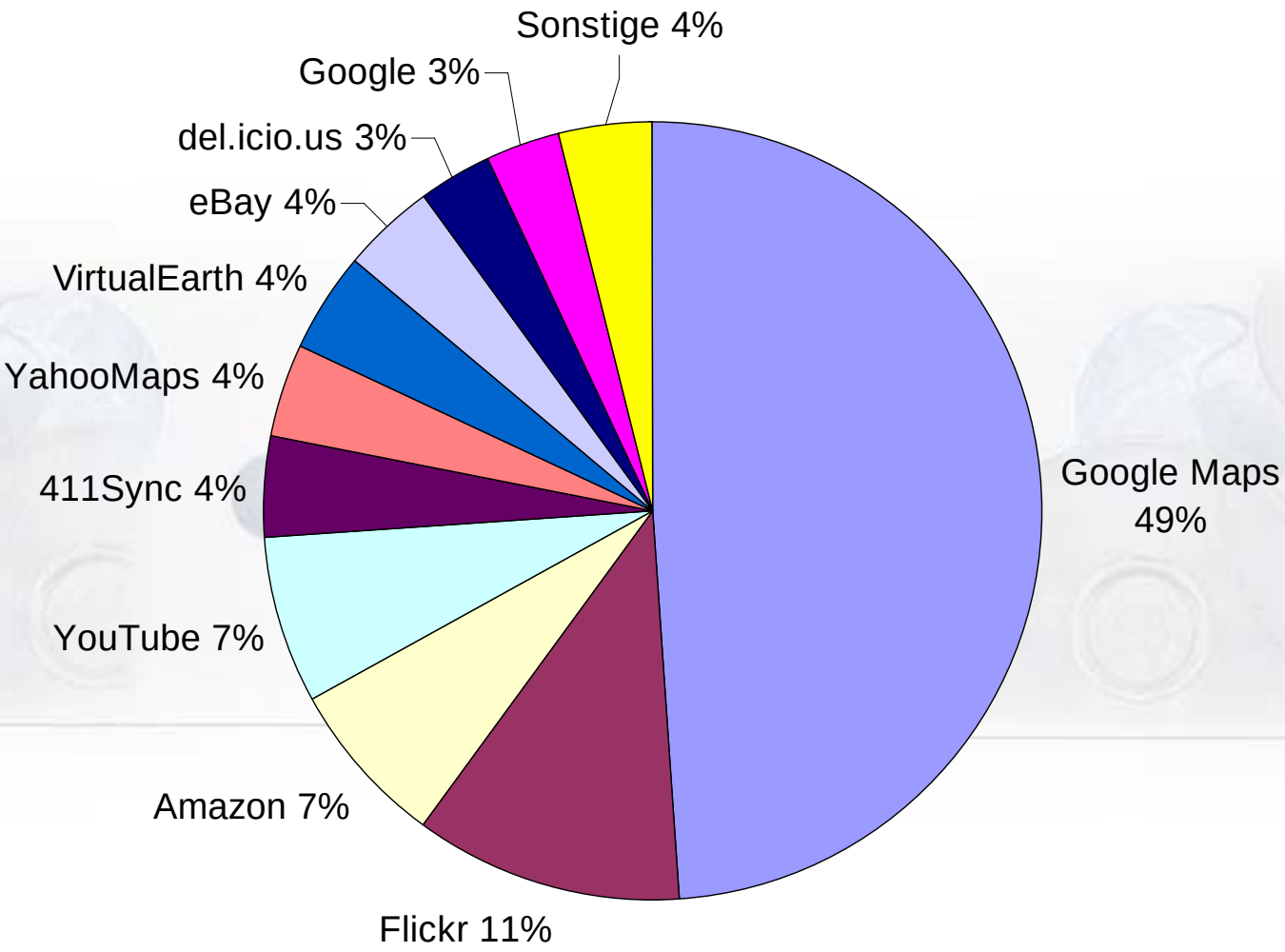
Vorteile:

- Sehr einfach.
- Nicht an bestimmte Schnittstellen gebunden. Zugriff auf SOAP, REST, etc. möglich.
- Viele Mashup-fähige Anwendungen verfügbar.
- Leichte Einbindung von SaaS (Software as a Service) Anwendungen möglich.

Nachteile:

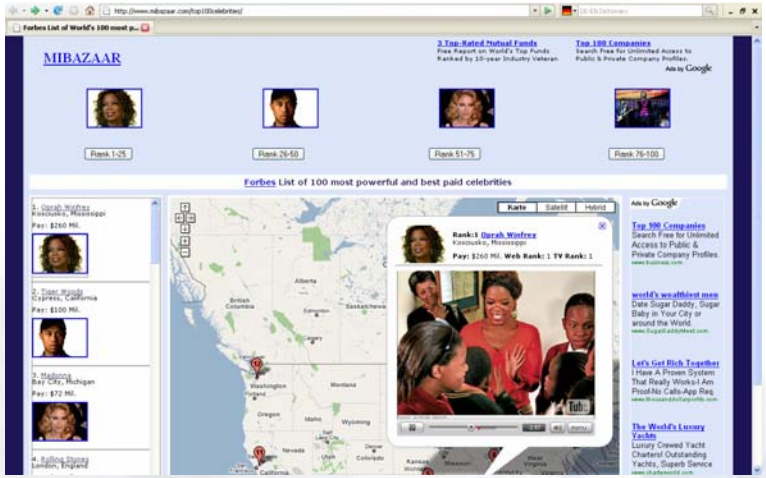
- Funktionalitäten nicht transaktionsbasiert miteinander verbunden.
- Logischen Abhängigkeiten schwierig zu realisieren.
- Schwierigkeiten bei der Abbildung komplizierter Prozesse.
- Keine Orchestrierung.

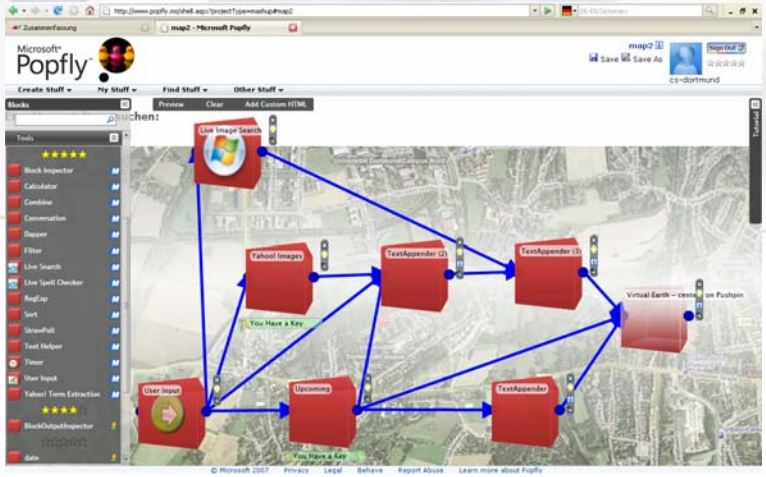
Verwendete APIs für Mashup



Quelle: programmableweb.com 27.11.2007

Beispiele Mashup





Ende

Vielen Dank für Ihre Aufmerksamkeit.

Haben Sie Fragen?

jan.gellweiler@fernuni-hagen.de