Institut für angewandte Informationstechnologie InIT

Visualisierung von Transponder-Daten mittels Mashup

Mitarbeiter:

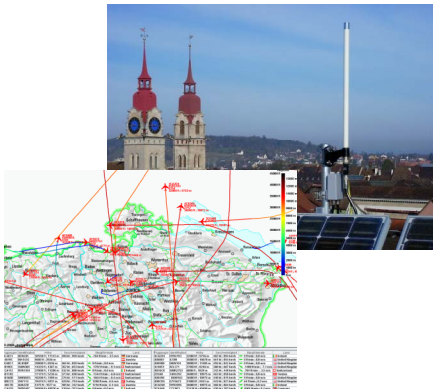
- Michael Jäger
- Andreas Loeber
- Daniel Kramarz
- Marco Vergari
- Prof. Dr. Marcel Rupf
- Prof. Dr. Karl Rege

Übersicht

- Geschichte
- Motivation
- Technologie
- Architektur
- Backend
- Frontend
- Herausforderungen
- Ausblick
- Fragen

Geschichte

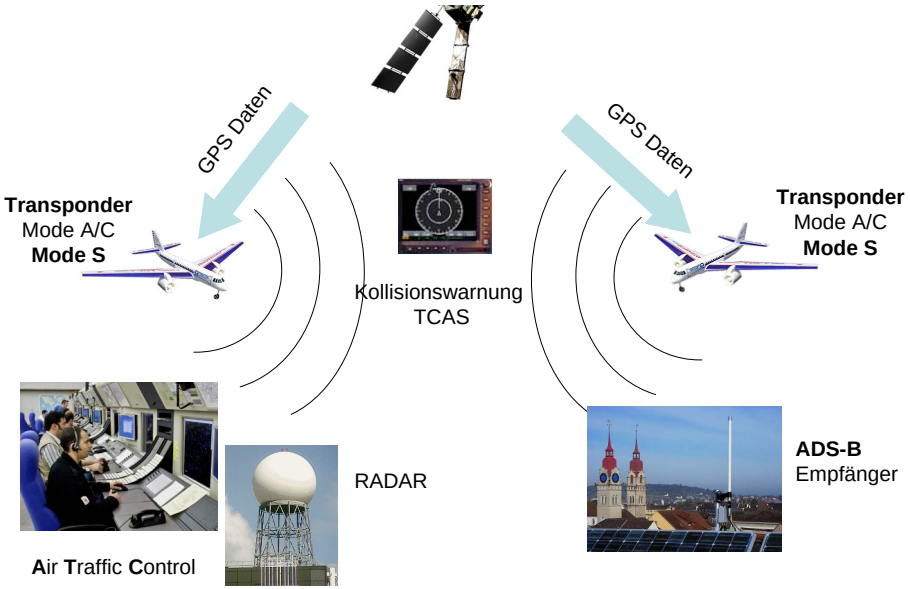
- Vorgängerprojekte
  - ADS-B Empfänger (Gruppe M.Rupf) mit einfacher lokaler Visualisierung auf PC
  - **I**nternet **I**nformation **B**roker: "Crawler/Datensammler für Comparis entwickelt)
  - Machbarkeitsstudie: Darstellung auf Google Maps als Semesterarbeit



Motivation und Herausforderungen

- Kombination verschiedener *aktueller* Technologien
- Google Maps **Mashup** - mit unterschiedlichen (verteilten) Datenquellen
  - Echtzeit Daten von eigenem ADS-B Empfänger
  - Frei verfügbare Daten aus dem Web via Internet Information Broker (IIB)
  - Kostenpflichtiger Web Service (SOAP)
- Weitere Herausforderungen
  - **Echtzeit** Datendarstellung in Google Maps (noch eher selten gezeigt)
  - Serviceorientierte Architektur (SOA) über Organisationsgrenzen hinweg
  - Eigentlich nicht geplant: Betrieb einer Ajax basierten Google Maps Anwendung

Technologie: Transponder und ADS-B



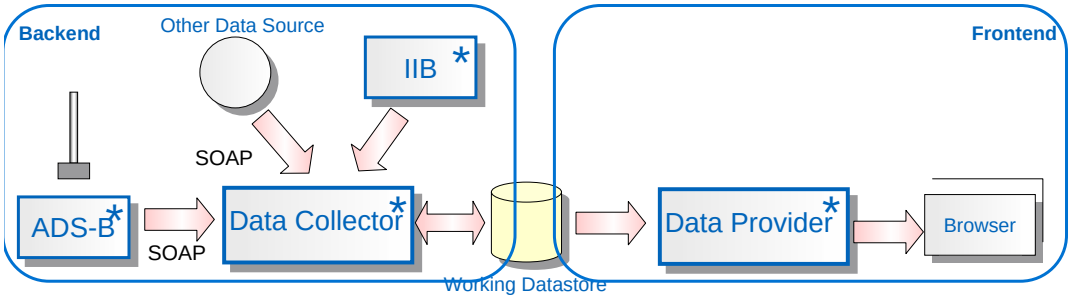
Technologie: eingesetzte Software

- Programmierumgebung: Java 6, .NET 2.0
- Web Server: Tomcat, Apache
- Kommunikationsprotokoll: SOAP
- Datenbank: MySQL
- Datenbeschaffer: IIB (basierend auf XSLT, XML, .NET 2.0 …)
- Datenbankzugriff: JPA (Java Persistence API)
- Browser Interaktion: AJAX/JSON
- JavaScript Bibliotheken: prototype und script.aculo.us
- Kartendienst: **Google Maps**

Alles basierend auf frei verfügbarer und/oder Open-Source Software

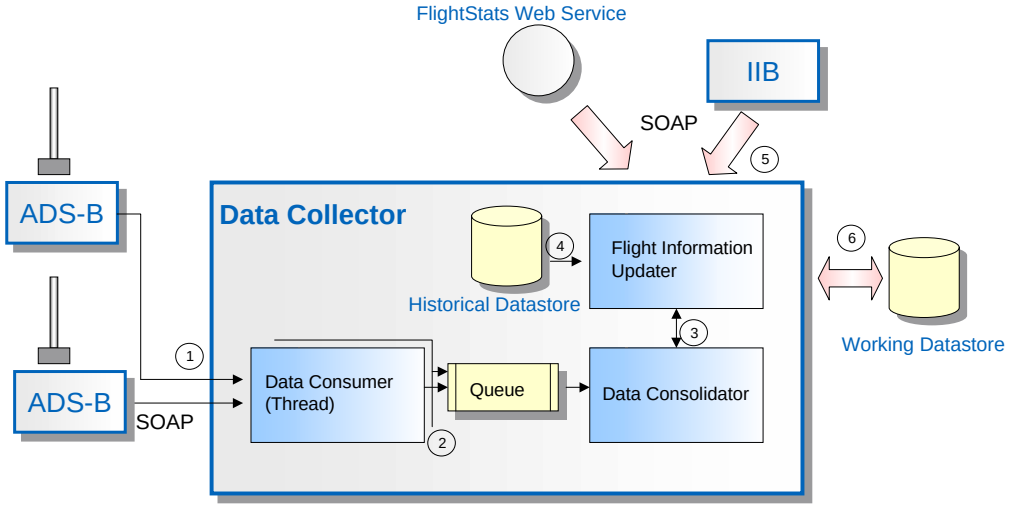
Architektur

- Data Collector: Sammelt Daten und sie bereitet auf
  - ADS-B: Daten als Web Service
  - Internet Information Broker: Für beliebige Daten frei aus dem Internet
  - Web Services von kommerziellen Anbieter
- Working Datastore: Datenbank zur Entkopplung
- Data Provider: Bereitstellung der Daten für Klienten



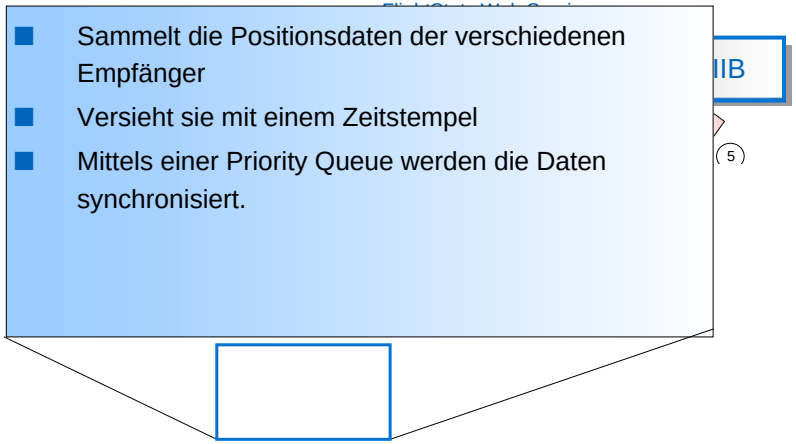
* an der ZHAW Entwickelt

Backend: Data Collector - die Teile



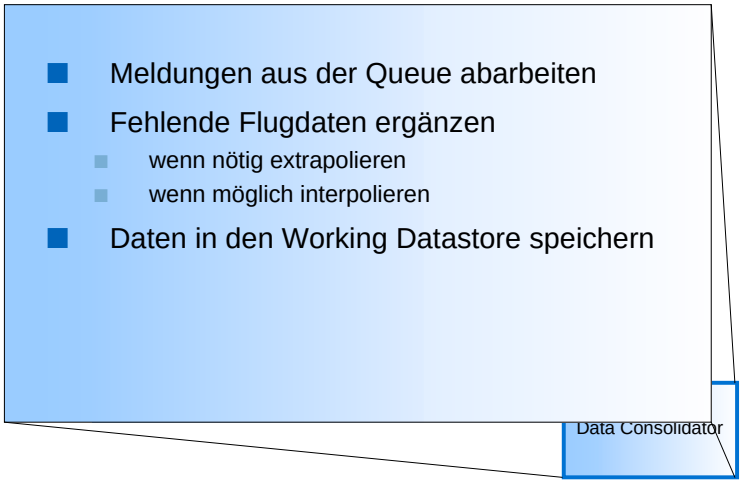
Data Consumer - im Detail

- Sammelt die Positionsdaten der verschiedenen Empfänger
- Versieht sie mit einem Zeitstempel
- Mittels einer Priority Queue werden die Daten synchronisiert.



Data Consolidator - im Detail

- Meldungen aus der Queue abarbeiten
- Fehlende Flugdaten ergänzen
  - wenn nötig extrapolieren
  - wenn möglich interpolieren
- Daten in den Working Datastore speichern

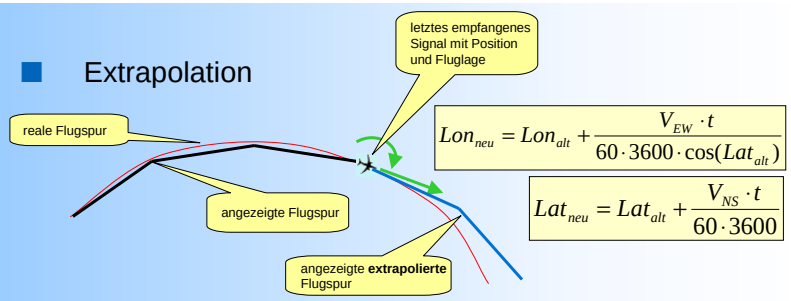


Data Consolidator - fehlende Flugdaten

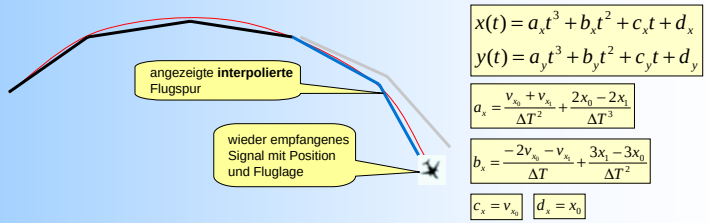
- ADS-B "Telegramme" mit Flugdaten und/oder Fluglage ca. alle 250 ms
- Problem:
  - Es müssen mehrere "Telegramme" empfangen werden, damit die vollständigen Daten vorliegen.
  - Der Empfang kann wegen topographischen Hindernissen zwischenzeitlich (oder vollständig) unterbrochen sein
  - Für die Darstellung ist es aber notwendig, dass die Daten kontinuierlich (im Sekundentakt) geliefert werden können
- Lösung für Fluglage: Extrapolation und Interpolation der fehlenden Flugdaten

Data Consolidator - Extrapolation und Interpolation

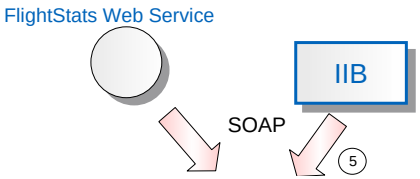
- Extrapolation



- Interpolation



Flight Information Updater - das Mashup

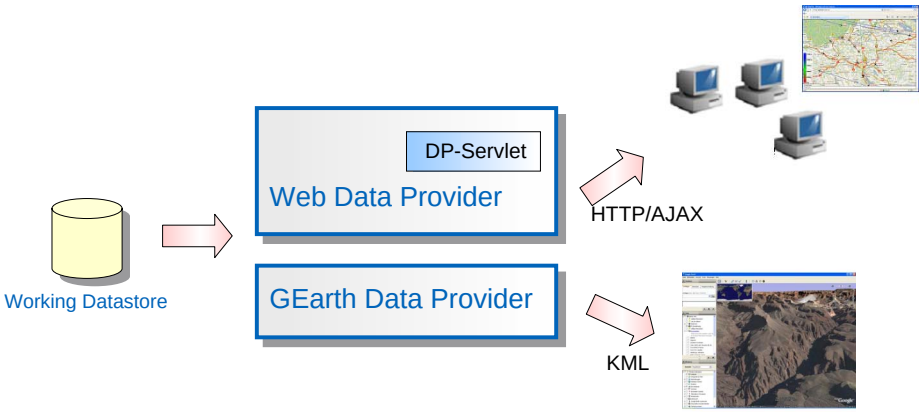


- **Reichert Daten aus unterschiedlichen Quellen an**
  - Datenbank: früher erfasste Daten
  - Live: mit Hilfe von kommerziellen Datenanbieter (Flightstats)
  - Live: mit Hilfe von Internet Information Broker (IIB)
- **Hier wird das eigentliche Mashup durchgeführt**
  - Alle Daten werden via Web Services (SOAP) bezogen
  - Problem: Assoziation der Daten bei unabhängigen, heterogene Datenbeständen
    z.B. ICAO, IATA

Internet Information Broker (IIB)

- **Extrahiert Daten aus dem Internet**
  - Der IIB ist ein Crawler, der die Information aus beliebigen (HTML basierten) Web Seiten extrahieren und als Web Service (via SOAP) zur Verfügung stellen kann
- **Gesteuert durch einfach zu erstellende XSLT Skripte**
- **IIB-XSLT-Builder als komfortables Werkzeug zur Erstellung dieser Skripte**

Frontend: der Data Provider

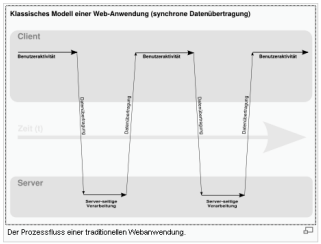


Technologie für Data Provider: Google Maps

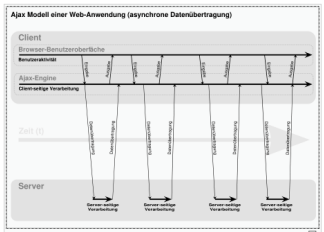
- Browser basierter weltweiter Kartendienst
  - Strassenkarten und Satellitenbilder
  - sehr gute Qualität, verschiedene Auflösungen
  - sehr günstige Bedingungen für **nicht kommerzielle** Anwender
- Eigene Daten lassen sich programmatisch (via Java Script) hinzufügen
- Google Java Script Bibliothek stellt verschiedene geometrische Primitiven zur Verfügung
  - Markers (für Flugzeuge)
  - Polylines (für Flugspuren)
  - Custom Overlays (für Schatten, Beschriftungen und Flughafen-Symbole)
  - …
- Jedoch bisher meist für Anreicherung mit statischen Daten verwendet z.B. Immobiliendaten, Standorte von Verbrechen, ...

Technologie für Data Provider: Ajax

- Traditionelle Web Anwendung
  - HTML Seite wird geladen
  - Daten werden in Formular eingegeben
  - Daten werden zum Server geschickt
  - nächste HTML Seite wird geladen, usw.
- Ajax Modell
  - HTML Seite wird geladen
  - Daten werden in Formular eingegeben
  - schon während/nach der Eingabe werden die Daten via Java Script zum Server geschickt und validiert
  - Vorteile
    - verbesserte Benutzerfreundlichkeit
    - in Summe weniger Datentransfer
  - Nachteile
    - verschiedene Technologien beherrschen
    - Unterschiede der Java Script Impl. im Browser



Der Prozessfluss einer traditionellen Webanwendung.



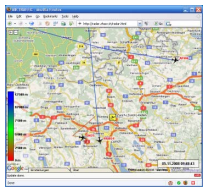
Der Prozessfluss einer Anwendung, wie er sich bei Ajax darstellt.

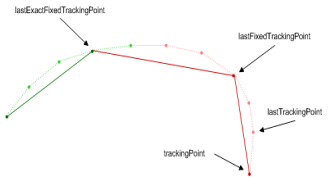
Technologie für Data Provider: Kombination

- Kombination von Google Maps mit Ajax
- Flugspuren
  - werden mittels Ajax vom Server nachgeladen
  - als Google Maps Polylines dargestellt
  - Flughöhe bestimmt Farbe; standard Polylines unterstützen keine Farbverläufe
- Performance im Browser (beim Client)
  - leidet stark je mehr Polyline-Fragmente auf der Karte eingezeichnet werden (nur ca. 200 Fragmente möglich)

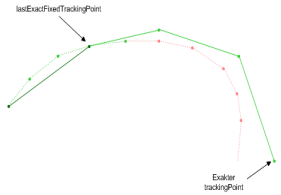
Probleme: Performance im Browser

- Flugzeuge (+Flugspuren) werden entfernt sobald nicht mehr im Sichtfeld
- Anzahl Stützpunkte für Polylines abhängig von Krümmungsradius





- Flugspuren werden inkrementell aufgebaut
- Extrapolierte Punkte setzen und später ersetzen durch interpolierte



- NEU von Google: Server Side Rendering (aber dadurch höhere Anforderung an Server/Netzwerk)

Server Performance

- Viele (Ajax) Anfragen an Server (mehrere 100 pro Sekunde)
- Zur Zeit auf einem einzelnen Server
  - 2 Xeon Quadcode 1.86 GHertz, 8 GB Memory, 4 *300 GB SAS RAID
- Massnahmen:
  - Cluster Architektur mit 4 VMs:1 Apache Frontend, z.Z. 2 Tomcat Backends, 1 Data Collector und DB
  - Trennung von statischen und dynamischen Daten -> weniger Last auf Backend
  - Vereinheitlichung der Anfragen, Frontend-Caching kann greifen
  - ....
- Hohe Schwankungsrate (kurzzeitig 80% der Schulnetz-Bandbreite)
  - Traffic Shaper, mod_bw und adaptive Updaterate (noch nicht implementiert)

Weitere zu lösende Probleme

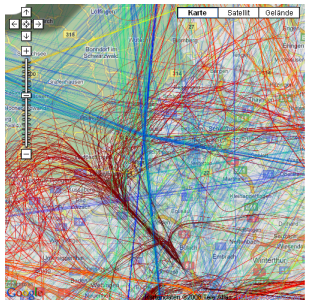
- Nicht alle Flugzeuge mit Mode-S Transponder ausgerüstet (ca 80%)
- Erkennung von Mode-A/C Signalen und Triangulation mit 3 Empfängern (DA 08)

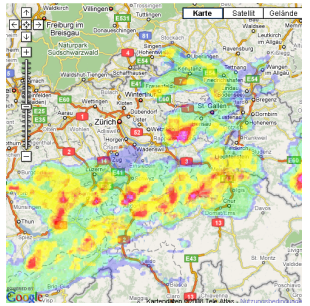
- An gewissen Orten kein Empfang (z.B. Südanflug)
- Lösung: zweiter besser positionierter Empfänger (z.B. ETH Hönggerberg)

- Zuordnung der Daten für Mashup (ICAO - IATA Problematik)
- Lösung: Flugpläne und bessere Heuristiken

Ausblick

Tagesverlauf Flugspuren





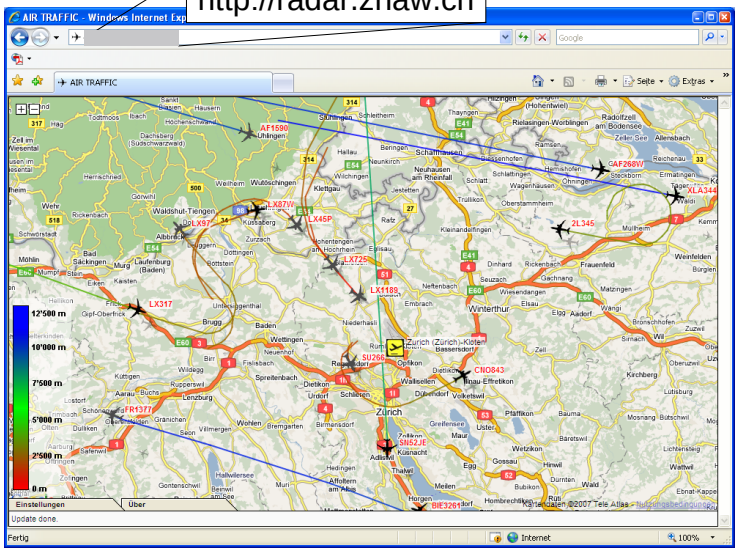
Wetter
Regen / Wind

ZH GIS
als Google
Maps
Overlay

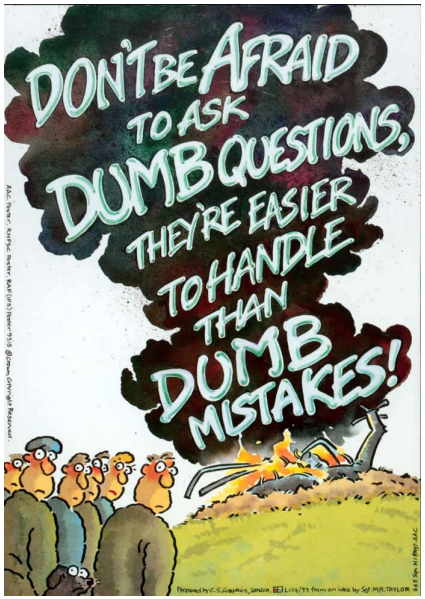
Globaler Flugverkehr
Technorama

Web-Applikation - Demo

http://radar.zhaw.ch



Fragen?



oder später auch

karl.rege <at> zhaw.ch