E-Government for Distributed Autonomous Administrations

Daniela Feldkamp, Wolf Siberski, Barbara Thönssen, Holger Wache
Stanford University, 26-28.03.2008
E-Government in Switzerland

Switzerland has a very long tradition in federated democracy

- 1 government
- 26 cantons (major districts)
- More than 2700 municipalities
- ... and all are real (!) autonomous!!

Question: How can we let them work together?
Solution so far

eGovCH – die E Government Architektur Schweiz

26 cantons
> 2700 municipalities
An example for a cross-administration process
Problem

- A huge group of autonomous municipalities providing services
- Agreed to cooperate, i.e. for cross-administration processes
- But reject any proposal which reduce their autonomy, i.e. any central instance modeling, hosting, maintaining or/and enacting cross-administration processes
The Vision

- No need for modeling, hosting and maintaining of cross-administration processes.
- First step: make the local services available for invocation by others.
- No central point of service registration. All services are registered at a local instance.
- Interaction between different autonomous units based on a minimum of commitments.
  - A very small ontology describes the subject of services with their minimal set of inputs and outputs.
  - The ontology is shared by all units.
Requirements

Every public administration
- remains responsible for its internal services
- is free to add or remove internal services from cross-administration service provision at any time
- is considered in any case a service is requested
- is able to execute its internal processes in its way
- is able to distribute their services to other public administration for execution.
Service-oriented Architecture

- **Service Publication**: The service provider publishes the service specification at a registry so that it can be found later on.
- **Service Lookup**: The service client searches for suitable services based on some constraints.
- **Service Invocation**: The service client executes the service.
Service Lookup: Ontology

- Ontology defines
  - Types of services
  - Service standard data
- Service description
  - Type
  - Location
- Each municipality share this ontology
- Every municipality is allowed to extend (NOT change) the ontology
- Ontology is already defined
Service Invocation: How to?

- Distributing a service request
  - To all municipalities: too expensive
  - To a central instance: not accepted
    - All services need to be registered there
    - No maintenance
Each municipality has its own representative: a broker

- Broker enact services of its municipality depending on their implementation
  - Workflow
  - E-Mail
  - ...

- Broker receives, distributes service requests and invokes services
- Broker are connected in a peer-to-peer network
Service Invocation:
Peer-to-Peer network

- Efficient distribution of service requests
- Broker = peer
- Broker are organized in a ring with shortcuts
  - Max. log n hops for service request distribution
- Note: one service request can invoke several services
  - At village level
  - At level of cantons
  - At level of government
- Service request forwarding with the help of a distributed Hash table
Service Invocation: Procedure

- Service request distribution in two steps:
  1. Find (smallest) superset of affected brokers
  2. Check with local, extended ontology if service has to be invoked
Service Invocation:
1. Distributed Hash Table

- First step: distributed hash table
  - Dimension: Service type and location
- Problem: Exact matches in DHT only
  - No match for "Logging" requests
- DHT needed to be extended with derived knowledge
- Derived knowledge has only to guarantee that affected broker will be elected
- Solution: derive all knowledge in advance
  - Similar to RDF databases
Service Invocation: 2. Instance retrieval

- Second Step: Instance retrieval
  - Classical description logic inference
  - Efficient because less expressive ontology required
    - only service type taxonomy
    - with some attributes for service data
- Needed because every broker can have its own extended ontology
Conclusion

- Huge autonomy without any central authority prevents SOA
  - Did you ever looked at a public UDDI repository?
- Distributed SOA: Broker
  - Service lookup: Minimal but extendable ontology
  - Service invocation: Peer-to-peer network
    - Efficient distribution of service requests
    - Distribution in two steps
Future work

- Convince the municipalities in Switzerland 😊
- Reliability
- Security

- Completing service data direct before invocation
  - Requires service invocation planning