Interaction Styles for Service Discovery in M-Business Applications

University of Ontario, Institute of Technology

March 9th, 2007

Markus Aleksy, University of Mannheim, Germany
Overview

- Introduction
  - SALSA-Project
  - Entities of the Context-driven Service Discovery Architecture
- Configurations
  - Basic Configuration
  - User-Managed Linear Configuration
  - Client-Managed Linear Configuration
  - Client-Mediated Linear Configuration
  - Server-Managed Hierarchical Configuration
  - Limitations and Capabilities
- Framework for the Design of Mobile Client Applications
- Conclusion
Introduction

SALSA-Project

SALSA: Software Architecture for Location-specific Transactions in Mobile Commerce

• Funded by Landesstiftung Baden-Württemberg

• Interdisciplinary project
  Involved areas: business administration, information systems, and computer science

• Follow–up project funded by German Research Foundation
Introduction

SALSA-Project
Service Discovery Architecture
Introduction

Entities of the Context-driven Service Discovery Architecture

The **Mobile Client** component is a client application, built using the SALSA client framework, running on a mobile device:

- The framework is responsible for mediating the communication with service brokers and service providers, and handles most of the context sensing. Data like the position, calendar information, device capabilities and profile data is gathered automatically.

- The mobile client offers a generic interface to the mobile user to query services and for displaying query results in the form of service descriptions.

- The client application adds implicit client-detected context information to the user’s service discovery request if available and explicitly allowed by the user.
Introduction

Entities of the Context-driven Service Discovery Architecture

- The Service Discovery Service (SDS) is a service broker that stores detailed descriptions of services that have been explicitly registered in its service repository.

- Service Orchestration Engine (SOE) is a control component that coordinates the interactions with multiple, lower-level SDSs in a nested architecture.
  
  - SOE’s usually exists within SDSs but they can also be contained by client applications. A SOE coordinates service invocations and returns their merged results according to the Facade Pattern. Additionally the SOE is able to invoke any service as a proxy for other components.
Introduction

Entities of the Context-driven Service Discovery Architecture

- **Service Providers** offer a service which may be used by any service requestor.
  - A service may be an electronic service, e.g. a web service based gastronomy guide or a non-electronic service, e.g. a bar or café. Service providers may furthermore offer lower-level service brokers that offer a certain kind of service using SDS technology, for example the gastronomy guide. An SDS is also a form of service provider.

- **Context Provisioning Services** (CPS) deliver implicit contextual information or they derive new contextual information from existing information for the service discovery process.
  - These services can be integrated into the mobile client application or the SDS.
Configurations

Basic Configuration

The basic configuration of our service discovery architecture follows the principle of service-oriented architectures represented by a triangle between the service requestor (mobile client), service broker and service provider.

- It is described by the Lookup pattern.
- Service providers are registered at an SDS. The mobile client sends a service request augmented with available context information from client-side CPSs to the SDS which can add further context from server-side CPSs that are available. The SDS returns the most suitable service descriptions corresponding to the request and context from its repository to the mobile client. Finally, the mobile client directly invokes one of the returned service providers using the service descriptions from the SDS. If the chosen service is an electronic service it is invoked over the Internet and the user receives the delivered value.
Configurations

Basic Configuration
Configurations

Possible Configurations are:

• User-Managed Linear Configuration
• Client-Managed Linear Configuration
• Client-Mediated Linear Configuration
• Server-Managed Hierarchical Configuration
Configurations

User-Managed Linear Configuration

• The user submits his desire to the client software that acts as the service requestor and sends requests to the USDS.

• The USDS as service broker returns a list of service descriptions of lower-level specialized SDSs which are presented to the user by the client software as services.

• The user then chooses an SDS according to his personal preferences and trust in the provider.

• The request is sent again to the chosen SDS which returns a suitable description of a service provider that is presented to the user by the mobile client application.

This interaction could also be performed over more than two levels since lower-level SDSs may return further specialized SDSs.
Configurations

User-Managed Linear Configuration

- User
  - Desire → List of Services
  - Choose Service → Result

- Mobile Client
  - Service Request → Service Descriptions
  - Service Invocation
  - Service Result

- USDS
  - Register Service

- SDS
  - Register Service

- Service Provider
Configurations

Client-Managed Linear Configuration

- This configuration, that takes the responsibility of choosing between the USDS returned service descriptions from the mobile user.

- The service descriptions that are returned by the USDS are processed automatically by the SOE of the client application, which sends requests related to the user’s initial requirements to all listed SDSs and returns to the user a merged result set. In this case the client application’s SOE has to detect duplicates which could occur from similar SDSs and filter them. This can be difficult due to the restrictions and limitations of mobile devices, primarily due to the multi-source nature of the data. Furthermore the client application has to connect to an unknown number of lower-level SDSs over the wireless channel.
Configurations

Client-Managed Linear Configuration

User → Desire → Mobile Client → Service Request → Service Descriptions

Service Provider → Register Service → SDS

SOE

USDS

Service Result → Mobile Client

Service Result → Service Provider

Service Request → SOE

Service Request → USDS

Register Service → SDS

Register Service → SDS
Configurations

Client-Mediated Linear Configuration

• This configuration is similar to the User-Managed Linear Configuration. The difference is that the mobile client application is customized by software components downloaded from service providers and executed by the component framework. In all the other configurations a generic client application displays service descriptions as service results. In this configuration, however, the additional downloaded components give the possibility to add more complex user interfaces for domain-specific services and delivery of additional information by services.

• Service providers can also offer components for realizing their security mechanisms. This mechanism even allows parts of the business logic to be downloaded to the mobile clients.

• Furthermore a downloadable component may improve the Quality of Service in scenarios with intermittent network connections.
Configurations

Client-Mediated Linear Configuration

- User
  - Desire
  - List of Services
  - Choose Service
  - Result

- Mobile Client
  - Component
  - Service Request
  - Service Descriptions

- USDS
  - Register Service

- SDS
  - Register Service
  - Service Invocation
  - Client components
  - Service Request
  - Service Result

- Service Provider
  - Register Service

Service Request
Service Descriptions
Configurations

Server-Managed Configuration

- In this configuration the USDS itself connects to lower-level SDSs using its SOE. The SOE submits requests to the other SDSs and merges the results sent back as a set of service descriptions to the client software which presents them to the user. In this configuration the detection and elimination of duplicates is much more efficient due to the higher computation power of servers.
Configurations

Server-Managed Configuration
Configurations

Limitations and Capabilities

• Bandwidth Usage and Latency
• Mobile Devices
• Java Environment
• Pricing
• Accuracy and Completeness
• Privacy
• Client Software Complexity
• User Experience
• Scalability
## Configurations

### Limitations and Capabilities

<table>
<thead>
<tr>
<th></th>
<th>User-Managed</th>
<th>Client-Managed</th>
<th>Client-Mediated</th>
<th>Server-Managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>0</td>
<td>--</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Latency</td>
<td>--</td>
<td>-</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Mobile Phones</td>
<td>--</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CLDC</td>
<td>+</td>
<td>+</td>
<td>--</td>
<td>+</td>
</tr>
<tr>
<td>CDC</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Pricing</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Completeness</td>
<td>--</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Privacy</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Complexity of Software</td>
<td>+</td>
<td>--</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Complexity for User</td>
<td>--</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Scalability</td>
<td>--</td>
<td>--</td>
<td>+</td>
<td>++</td>
</tr>
</tbody>
</table>
Framework for the Design of Mobile Client Applications
Framework for the Design of Mobile Client Applications

Design of our Framework

Our framework for context-sensitive mobile applications consists of:

• **Service Discover Component**

• **Context Manager**
  The Context Manager aggregates information from the context sensors, like position information, user profile data or free time slots. It offers a convenient interface to the service discovery component and other application for access to context information.

• **Component Manager**
  The Component Manager offers registration and download of components during runtime. It manages the life-cycle of those components and tries to alleviate the memory constraints of mobile devices by having only relevant components in memory.
Framework for the Design of Mobile Client Applications

Design of our Framework

• **Generic Communication Framework**
  A generic ORB offers an abstraction from the used RPC protocols and the used framework. It manages the connections to the service providers and the broker. Multiple protocols could be added to the broker via the Component Manager during run-time.

• **Adaptable User Interface Framework**
  The User Interface adaptable to the current context and the currently used service to offer the best possible usability with the limited capabilities of the current mobile devices.
Framework for the Design of Mobile Client Applications
Framework for the Design of Mobile Client Applications
Conclusions

• We have presented a highly-flexible, context-driven service discovery architecture and explained the configurations in which it can be used.

• We have evaluated the pros and cons of each configuration and explained their interaction characteristics from the client perspective.

• We also discussed the characteristics of each configuration from the perspective of different requirements.

• For the SALSA project, which focuses on scenarios in which the user wants to find a previously unknown service in an ad hoc way depending on the current situation, we have identified the optimal configuration as the Client-Mediated Configuration, since we rely also on CDC as the mobile client programming platform.
Conclusions

• This configuration has only minor drawbacks and it offers a very good user experience, better offline functionality and a better user-service interaction. In the event that CDC is not available the choice depends on other factors. In particular, it can not be assumed that users will trust all SDSs or service providers - a fact that makes Server-Managed Configuration an unsuitable choice.

• In general, the requirements and needs of the scenario in hand will determine which configuration is the most suitable. The key advantage of the architecture is that the basic components can be used to support any configuration in a flexible way to best meet the prevailing requirements and environment.

• Our framework enables the easy usage of context without implementing the context matching and the context all over again in any component and having the necessary components always available on the device. Furthermore the UI components are able to adjust themselves to different types of devices or situations without having code for complex detection of devices capabilities in the applications.