An Integrated Platform for Location-based Services Built on Java Technologies

C. Kassapoglou Faist
CSEM
CSEM

Centre Suisse d’Électronique et Microtechnique

Technologies for innovation

- High technology know-how transfer to the industry
- I am with the Advanced Systems Engineering Division
Outline

- Motivation
- PoLoS project
  - End-user perspective
  - Operator perspective
- PoLoS and J2EE (JBoss)
Motivation

Location-based services: quite promising, but their growth has been rather slow
- too complex (several fields, many actors)

Objective:
A tool for easy and low-cost development and provisioning of location-based services (LBS)

Use of open, standard technologies
Outline

- Motivation
- PoLoS project
  - End-user perspective
  - Operator perspective
- PoLoS and J2EE (JBoss)
PoLoS (EU, IST project, 2002-2004)

PoLoS: An integrated platform for the creation and provision of location-based services (LBS)

- an execution environment for LBS
- a development tool, for high-level specification, deployment and testing of new services
Partners

- University of Athens (GR)
- Alcatel SEL AG (DE)
- CSEM (CH)
- Intracom S.A. (GR)
- EPSILON S.A. (GR)
- Telefonica I+D (ES)
- Epsilon Consulting Ltd. (CY)
Service Execution Flow

Geo Information System

WLAN (Indoor)

GPS

Platform

Service Execution

Position

Mobile (Outdoor)

Service
Service Creation
Features (I)

- Generic with respect to LBS types (no service-specific components)
- Portable: PoLoS is independent from specific hardware
- Independent from end-user terminal technologies: supports WAP, SMS, HTTP
- Multiple positioning techniques: GPS, GSM- and WLAN-based (simultaneously)
- Flexible with respect to external components
Features (II)

- Support for several LBS execution models: client pull, server push, time and event scheduling
- Service specification in a high-level, specially designed, XML-based language
- Easy service specification and handling using the Service Creation Environment (graphic and text edition, deployment, debugging)
- Service and user management.
- Optional access control at per user or per service basis.
LBS Scenarios

**Typical scenarios for LBS**
- getting the exact address of one’s current location (“Where am I?”),
- finding a nearby point of interest (“What is the closest cinema?”),
- obtaining navigation directions (“How do I get there?”),
- tracking other users (“Where is my friend?”, “Notify me when one of my friends is close”),
- fleet management applications,
- zone-based advertising (“Welcome to our site!”), etc
PoLoS Use (1):

Invocation and result of the proximity service using WAP
PoLoS Use (2):

Invocation and result of the proximity service using SMS
PoLoS Use (3):

INVOCATION (FROM THE SERVICE CUSTOM JSP):

TEXT RESULT:
at 0 meters Go Straight to I9 ; at 0 meters turn right to IHall 1 ; at 20 meters go straight to IS1 ; at 10 meters turn right to S1 ; at 0 meters turn left to 0 ; at 0 meters Go to end ; at 0 meters Go Straight to S1 ; at 0 meters go straight to AIS1 ; at 10 meters turn left to AHall 1 ; at 0 meters turn right to A22 ;
PoLoS Architecture
PoLoS Kernel Functionality

- Deployment of new services
- Service bookkeeping
- Invocation of services
- Execution of services
- Availability of peripheral components (POS, GIS, interfaces), task scheduling and other functions to service executables
- User registration
- Other (billing, …)
Kernel Architecture

1. Service/User Registry
2. Service Invocation Module (SIM)
3. Service Deployer
4. Compiler
5. Service Execution Environment
6. End-user Interface
7. Interface to Positioning Technology
8. Interface to GIS Data
9. Schedule

Interface to Service Creation Environment
Technologies (1)

- PoLoS platform: an application server, based on **Java 2 Enterprise Edition** technology (*jboss*)
- Services: instances of **Enterprise Java Beans**
- GIS, POS components: available to the services as **Enterprise Java Beans**
- New service deployment and debugging: **Web services** (*Apache axis*) offered by the platform
- GIS, POS underlying systems: **Web services** (*Axis*) invoked by the platform
Technologies (2)

- All data exchanged in XML, even between internal components to the platform (use of SAX, DOM, XSLT and jdom)
- Standardized access to underlying network infrastructures using OSA
Positioning(I)

- Three positioning systems: GPS, GSM-based, WLAN-based
- When a service executes, the POS system used is:
  - either specified in the service logic
  - or determined based on the end-user address type
- Types of requests:
  - Request/Reply
  - Generic Request
  - Event-triggered notification (GSM- and WLAN- based)
  - (Periodic Request not implemented; use of the kernel scheduler)
Positioning (II)

- **GSM-based (outdoor):**
  - emulated positioning (on a real GMLC system) (Alcatel)
  - positioning system by Telefonica Moviles (for authorized subscribers)

- **WLAN-based (indoor):**
  - Location detection based on Nibble system (symbolic coordinates) (UoA)

- **GPS:**
  - Requires a proxy software on user’s GPS-enabled terminal
  - Location coordinates, piggybagged to the service invocation request, are stored to a repository
Geographic Information Systems

- **GIS services** (available to the platform as Web Services behind a session EJB):

<table>
<thead>
<tr>
<th>GIS services</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find Location</td>
<td>Location information corresponding to the supplied coordinates</td>
</tr>
<tr>
<td>(outdoor and indoor)</td>
<td></td>
</tr>
<tr>
<td>Proximity</td>
<td>Information on the closest POI (theater, …) to the supplied coordinates</td>
</tr>
<tr>
<td>(outdoor and indoor)</td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td>Navigation directions on how to go to a given point</td>
</tr>
<tr>
<td>(outdoor and indoor)</td>
<td></td>
</tr>
<tr>
<td>Geocoding</td>
<td>Address to coordinates conversion</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- PoLoS project
  - End-user perspective
  - Operator perspective
- PoLoS and J2EE (JBoss)
Service Specification

- Through a script (XML-based)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PolosXML>
  <service name="serviceName" ...>
    ...
  </service>
  <configuration>
    ...
  </configuration>
</PolosXML>
```

- Service logic control language (SCL) specially designed (usual programming language constructs and PoLoS specificities)
Service Control Logic Language

- Service entry points (->methods): `<entry>`
- Variables: `<set>`
  Support of usual arithmetic and logic expressions and string handling. Treelike variable structures through nested `<set>` elements.
- Invocation of platform components, other active services, native Java methods (attachments): `<invoke>`
- Control of logic flow: `<loop>`, `<break>`, `<if>`
- Error control: `<try>...<catch>`
- Persistent variables using repositories (service configuration (ro), service and service-user levels)
- Indirect addressing (based on variable names)
Service Logic Specification Example

```
<service name="GetMyLocation" >
  <entry label="main">
    <set name="userpos">
      <invoke component="POS">
        <set name="userid" value="parameters.userId" />
      </invoke>
    </set>
    <set name="useraddress">
      <invoke component="GIS">
        <set name="coordinates.north" value="userpos.latitude"/>  
        <set name="coordinates.east" value="userpos.longitude"/>  
      </invoke>
    </set>
    <set name="result.location" value="useraddress.textposition"/>
  </entry>
</service>
```
Service Creation Environment

An IDE for:

- Edition of LBS scripts
- Deployment on the PoLoS platform (securely)
- Testing of new services (tracing)
Technology

- Service Creation Environment based on Eclipse SDK.
- PoLoS menus, views and operations defined and implemented as plugins (service logic text edition and service deployment, graphic edition, debugging)
- New service deployment and debugging: Web services (Apache axis) offered by the PoLoS platform
- Service testing based on Apache log4j.
Text Editor
Visual Editor
Service Logic into an EJB

- Translation of service logic (in SCL) into a Java class, which is encapsulated into a stateless session EJB.
- Translator SCL->Java based on **JavaCC** and **JTB**
Translation phases

Preprocessing (parsing of XML)

Lexical & syntax analyses

Semantic analysis and intermediate code generation

Java code generation

```java
public class GetMyLocation implements Serializable {
    String serviceId = "GetMyLocation";
    // Entries
    public String entrymain (String parametersS) {
        ...
    }
}
```
EJB Class of a LBS service

```java
public class GetMyLocationEJB implements SessionBean {
    ...
    public void ejbCreate() throws RemoteException {
    ...
    public void setSessionContext(SessionContext sc) {
    ...
    public void ejbActivate() {
    ...
    public void ejbPassivate() {
    ...
    public void ejbRemove() {
    ...
    public String entryMain(String XMLInputParameters) {
    ...
    public String entryInit(String XMLInputParameters) {
    ...
    ...
}
```
Service Deployment on PoLoS

- Invocation of a secure Web Service (SOAP over HTTPS) exported by PoLoS platform
PoLoS Platform Management

PoLoS platform - Main Menu

Components
- POS
- SIM
- ServiceDeployer
- GIS

Interfaces
- Configuration Manager
- User Management
- Service Management
- Scheduler Manager

http://195.134.67.16:8080/polos/management/SRV_management.jsp
Service Testing(I)

- Post-mortem debugging (trace on service execution) based on log4j
Service Testing(II)

Event detail LINE 2: ENTRY POINT Main
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 2: Error in node variable name serviceId. Missing parameter.
  date: 2003.06.27 at 12:16:22 CEST
  level: ERROR

Event detail LINE 4: POS INVOCATION
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 9: GIS INVOCATION
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 9: Invalid XML data for variable creation
  date: 2003.06.27 at 12:16:22 CEST
  level: ERROR

Event detail LINE 14: Reading variable var from session repository. Value: null
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 14: Writing variable var to session repository
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 22: Reading variable var from config repository. Value: <var>"<?xml version="1.0" encoding="UTF-8"?>
<var>"</var>
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 22: Reading variable var from session repository. Value: <var>"<?xml version="1.0" encoding="UTF-8"?>
<var>"</var>
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 22: Writing variable var to config repository
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO

Event detail LINE 23: Reading variable var from global repository. Value: <var>"<?xml version="1.0" encoding="UTF-8"?>
<var>"</var>
  date: 2003.06.27 at 12:16:22 CEST
  level: INFO
Outline

- Motivation
- PoLoS project
- End-user perspective
- Operator perspective
- PoLoS and J2EE (JBoss)
J2EE Basic Ideas

Distributed Application Model:

Components in containers:

Separation of business logic, held in components, from implementation details, provided by containers.
J2EE Architecture

Three types of EJB: Session, Entity and Message-driven
PoLoS on Jboss (service execution)

14/06/2005
Web Services

Cross-platform, cross-language, distributed computing applications
- SOAP/ HTTP message exchange
- WSDL: XML-based language for service endpoint description

Axis: A SOAP engine
- Plugs into servlet engines (Tomcat)
- Extensive support for WSDL,
- generation of Java classes from WSDL
Conclusion

- PoLoS has been successfully demonstrated
- Extensive use of open and open-source technologies
- A very positive experience
  - Easy development, re-usability
  - Portability, scalability
References

PoLoS
• www.polos.org

J2EE
• www.jboss.org (open-source J2EE platform)
• www.java.sun.com/j2ee (doc, tutorial and open-source J2EE platform)
Thank you!