Service-ORiented Computing EnviRonment (SORCER)

Michael Sobolewski
sobol@cs.ttu.com
I do not believe traditional tools, technologies, and methodologies support

Distributed Megaprogramming, Service-Oriented Programming, or Self-Aware Computing
Presentation Agenda

- Distributed Self-Aware Service-Oriented Computing
- Federated S2S Environment
- Job/Task/Context/Method Paradigm
- SORCER Functional Architecture
- Design Issues (UML diagrams)
- Summary
Evolution of Computing

- Client-server silos
- Web-based computing
- Peer-to-Peer
- Service-to-Service
Object Space vs. Job Space

Data, Operations, and Control Strategy

Client
Request

Sender
Message

Requestor
Job (Program)

VM
Client
VM
Sender
VM
Requestor

Business Logic

Database

Server

Object Space
△ - mini-server (object)

Job Space (Federation)
○ - service provider (object space)
Self-Aware
Service-Oriented Computing

• A federation knows what is doing
• Learn from experience and adapt to surprises
• Is aware of its behavior and explain itself
• Is able to anticipate different scenarios and predict and plan for novel futures
• It would learn, not crash, when faced with a new situation
• Self-testing, self-debugging, and self-explaining within a federation
SORCER Vision

Federated S2S environment to ...

- Build new services
- Convert legacy apps to dynamic SORCER services (J2EE™ technology)
- Assemble SORCER services together (RMI, Jini, Rio technologies)
- Create modern clients accessing services
Megaprogramming Domain

System Design

Subsystem Design

Component Design
Many Kinds of Networks

<table>
<thead>
<tr>
<th>Client-Server</th>
<th>3/N-Tier Objects</th>
<th>Net Apps Legacy to the Web</th>
<th>Net Services The Computer is the Network</th>
<th>Next Network of Embedded Things</th>
<th>After that Network of Things</th>
</tr>
</thead>
<tbody>
<tr>
<td>100s</td>
<td>1000s</td>
<td>1000000s</td>
<td>10000000s</td>
<td>100000000s</td>
<td>1000000000s</td>
</tr>
<tr>
<td>NIS, NIS+ RPC, XDR</td>
<td>+ CDS + CORBA</td>
<td>+ LDAP (+) + CORBA, RMI</td>
<td>+UDDI + SOAP, XML</td>
<td>+ Jini + RMI/Jini</td>
<td>+ ?</td>
</tr>
</tbody>
</table>

Many Types of Services
SORCER Service: An entity that publishes (by attributes) functional capabilities on the network. (Mesh, Thermal Analysis, Print, etc..)

- DOE Services
- Analysis Services
- Optimization Services

Clients may not care where or who supplies the services

The computer is the network that exposes services to clients AWAT
Provider As a Network Object

Service Requestor

Service Provider

Mobile Proxy Object
Service-to-Service (S2S)

Service Providers

HTML/XML

Client1

Provider1

... Providerp

DBMS

HTTP

Object Registry/ Lookup Service

Method calls service(Exertion)
Applying NOO Techniques

- Service activity is a special object of type: *Exertion*
- Exertions are executed by network objects/service providers of type: *Servicer*
- Service providers form P2P environment
- Service is requested by calling the method: `service(Exertion)`
- Service providers are identified by a type with methods:

  ```java
  public ServiceContext selector(ServiceContext)
  ```
• All service activities implement this interface:

```java
public interface Exertion {
    // Apply this exertion method to the specified context
    public Exertion exert()
        throws RemoteException, ExertionException;

    ...
}
```
• All services implement this interface:

```java
public interface Servicer {
    // Put into action the specified exertion
    public Exertion service(Exertion exertion)
        throws RemoteException, ExertionException;

    // Monitoring methods
    ...
}
```
ContextMethod attributes: service type, selector, group, provider name, method type
Method type:  
preprocess,  
postprocess,  
append

TC – Task Context, CC – Control Context

Job  Task  Service Context  Service Method
Workflow vs. Job

Workflow
\[ W_0 = \{ (A1, A2), (A1, A3), (A2, A4), (A2, A5), (A3, A6), (A4, A7), (A5, A7), (A6, A7) \} \]

Task/Job
\[ J_0 = (T1, J1, J2, T2) \]
\[ J_0 = (T1, J1, J2, T2) \]
\[ J_1 = (A2, A3) \]
\[ J_2 = (A4, A5, A6) \]

Sequential relationship
Unidirectional aggregation
Inherent control strategy
Explicit all connections

Is-part-of relationship
Bidirectional aggregation
Control strategy separated
Workflow defined implicitly
JTCM Paradigm
Service Binding

Job as a Megaapplication

Method type: preprocess, process, postprocess, append
Federation of Services as a Job Runtime Environment

Method type: preprocess, process, postprocess, append
1. Update combustor PCS

2. Request for nozzle validation

3. Check for nozzle insertion

4. Perform modal analysis

5. Perform CFD blow analysis

(UG) (Blow Analysis) (ProE) (Blow Analysis)
Question

What does it mean to be a service?
A service is an act of requesting a service\text{(Exertion)} operation from a service provider.

If accepted
\texttt{then} exertion.exert() \texttt{else} forward to a relevant service provider
Dynamic Capability Trend

What we’re doing now with Jini

Jini Technology Based System
- Self Healing
- Dynamic
- QoS
- Network Centric

Traditional
- Robust
- Transaction Oriented
- Data Centric
- Host Centric

Where capabilities and products are today

Capabilities

Readiness

Target System X
Task Execution

- **Servicer**
- **Provider**
- **Remote**
- **Arithmetic Interface**
- **Unicast Remote Object**
- **Sorcer Joiner**
- **JavaSpace**
- **Provider Worker**
- **Service Provider**
- **Arithmetic Method**
- **Arithmetic ProviderImpl**
- **Remote ServiceMethod**
- **Entry ServiceTask**
- **Service Context**
- **ServiceTask**
- **0..* Has data**
- **Tasks**
- **Drops task**
- **Invokes**
- **Runs**
- **Submits task**
- **Publishes Proxy**
- **Submits task**
- **Requestor Runner**
- **Requestor**
Question

Does SORCER use service brokers?
A SORCER service broker is called a *jobber*. 
Job Execution

Service Servlet
- <<HTTP>>
  - Servlet Protocol
    - Proxy Protocol
      - Uses
        - Sorcer Launcher
          - Uses
            - JobBrowser

Jobber
- Provider
  - Requestor
  - Runner
  - Cataloger
    - Service
      - CatalogImpl

TaskDispatcher Factory
- Service
  - Provider
    - Service Access: Direct, Catalog, Space
      - TaskDispatcher
        - Catalog TaskDispatcher
          - Space TaskDispatcher
        - Drops job
      - TaskDispatcher
        - Service TaskDispatcher
      - Invokes
        - Requestor Runner
          - Submits job
            - JobberImpl
              - Uses
                - Catalog
                  - Uses
                    - Jobber
                      - Service Provider
                        - Provider
                          - Requestor
                            - Uses
                              - Arithmetic Requestor
                                - Creates
                                  - TaskDispatcher
                                    - Service TaskDispatcher
                                      - TaskDispatcher
                                        - Drops job
                                          - Service access: Direct, Catalog, Space
Rio

- Jini™ technology-based Service Beans ("JSBs") are the fundamental domain specific computational entities on the network
- Are provisionable based on their QoS attribute
- Jini technology-based Service Beans are instantiated by Cybernodes
  - Cybernodes run on computational resources
  - Cybernodes can contain multiple service beans
Service Delivery Network Grid

Self-organized service providers allocated to best resource

Dynamic QoS provisioning grid

Compute resource management grid
<table>
<thead>
<tr>
<th>Bootstrapping Type</th>
<th>Server Type</th>
<th>NDS</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>java SorcerJoiner -sProviderClass</td>
<td>RMI server (JRMP/IIOP)</td>
<td>JNDI/RMI Reg JNDI/LDAP</td>
<td>RMI/CORBA</td>
</tr>
<tr>
<td>java SorcerJoiner -pProviderClass</td>
<td>Service provider (Jini)</td>
<td>LUS</td>
<td>Jini</td>
</tr>
<tr>
<td>java SorcerJoiner -pProviderClass:ProxyClass</td>
<td>Service provider with smart proxy (Jini)</td>
<td>LUS</td>
<td>Jini</td>
</tr>
<tr>
<td>Provisioning (Rio)</td>
<td>JSB</td>
<td>LUS</td>
<td>Rio/Jini</td>
</tr>
</tbody>
</table>
SORCER Code Mobility has many forms

1. Proxies
2. Exertions
3. Task Methods
4. Agents
5. Service Beans (JSBs)
6. Service UIs
SORCER Runtime Environment

Domain specific:
- Providers
- Requestors
- ServiceUIs

Infrastructure:
- Jobbers
- Droppers
- Catalogers
- Persisters
- Notifiers
- Service UIs
- Websters
- Cybernodes
- Provisioners
- Lincolns
- Web Server/App Server

SORCER

Rio
SORCER’s C³

- **Service Centricity**
  - everything is a service, each represented as an object on the network identified by type

- **Network Centricity**
  - services discover each other
  - the service is the network (N-1, 1-1, 1-N, S2S)

- **Web Centricity**
  - Interportals/Extraportals to services with thin web clients (applets/servlets)
Summary (CNb)3

SORCER’s N³

- Co-location Neutrality
- Protocol Neutrality
- Implementation Neutrality

* Business logic
WCH/W – Who cares how/where?
Summary (CNb)3

Architecture Qualities $b^3$

- **Accessibility**
  - Web Centricity, standalone clients, agents
- **Adaptability**
  - Mobile Code
- **Scalability**
  - Network Centricity, Federated Services
Everyone Can Contribute
Pervasive SORCER.net

Knowledge Systems
Soft Computing
Data Persistence and Mining
Image Processing/Analysis
Computer Vision
Multimedia
Parallel/Space Computing
Large Scale Systems
Mobile Computing
Distributed Computing
Security
Mobile Computing

SORCER.net
Service Provider
Peers/Groups
Surrogates
Mobile Devices …
The Edge
Service Interface
Conclusion

- Jini™ and Rio technologies enable federated S2S, platform independent, real world megaprogramming environments.
- A SORCER job is a distributed megaapplication executed in a federated S2S environment.
Michael Sobolewski
sobol@cs.ttu.com