Good **Presentation** is like a mini-skirt

- **Short** enough to draw the attention
- **Long** enough to cover the subject
A literature survey on:

Service Tailoring approaches

Mohammad Zarifi Eslami
IS weekly seminar, 16 Feb. 2010
Presentation outline
Outline

16 February 2010                                Service tailoring approaches

U-care
Concern: Service tailoring
Survey Framework?
Tailoring methods
Some examples
Summary
Services layer for integrated home care systems

Provides tailorable, evolvable and non-intrusive home care services

Aims at supporting elderly to prolong independent living
Concern: Service tailoring
Some definitions

**Bonett:** Personalization involves a process of gathering user-information during interaction with the user, which is then used to deliver appropriate content and services, tailor-made to the user’s needs.

**Jakob Nielsen:** Customization, under direct user control, the user explicitly selects between certain options (explicit). Personalization, the user is seen as being passive, or at least somewhat less in control (implicit).

**U-Care:** Service tailoring process is a process of creating a new service that match individual needs of imbedded service users, by configuring and composing existing services. Service tailoring enable user to create personalized required services.
Different end-users (care-receiver, care-giver, technician)

Domain is closed environment

<table>
<thead>
<tr>
<th>Knowledge</th>
<th>User</th>
<th>Care receiver</th>
<th>Care giver</th>
<th>Tailoring technician</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextual</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>domain-specific</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>technical</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Different end-users have different knowledge and skills
Service tailoring methods
Survey framework?

User select his/her goals

Required services are selected from existing services

Configuration of individual services

Composition of configured services

User-centric composition
Service adaptation
Service personalization
Service configuration
Service composition
Dynamic service composition

Survey result comes from 31 related research works

16 February 2010  Service tailoring approaches
Some relevant tailoring methods

- Goal based
- Semantic based
- Context-aware
- Rule based
- Learning user behavior
- AOP

These methods are not mutually exclusive
Goal based

Has been used in different areas of computer science to identify stakeholder’s objectives, determine requirements for software systems and guide system’s behavior.

In service-oriented computing goals can be used to formulate users’ requirements.

Goals can be used to represent service requests and therefore, they can be used to guide service discovery and composition activities.
Semantic based

- Has been used to provide Machine-Understandable information
- Enables services or frameworks to exploit machine-understandable information to interoperate.
- Assist in automation of various aspects such as discovery, invocation and composition.
- Utilizing semantic similarities among components/services, improves adaptability in selection.

A Web service is an accessible application that other applications and humans can discover and trigger to satisfy multiple needs
Any information that can be used to characterize the situation of an entity (a person, place, or object)

Any information relevant to the interaction between a user and an application, including the user and applications themselves

Enables systems to create more dynamic, flexible and intelligent services

Enables delivering dynamic services due to dynamicity of user’s situation
Rule based

Control events and guide behavior of the system

Can be expressed as conditions for decision making

Can be used for configuration of individual services or composition

If A then B else C
Why learning based method:
(1) Predefined rules usually may not be allowed to modify once they are deployed
(2) It may be difficult to define a generic rule that is applicable to every user.

Why Rule based method:
It may not be possible to derive rules for new users/applications
We will discuss AOP with the related example if we have time.
Example 1: Context-aware approach
Example 1: Context-aware


(1) Allows not only designers but also end users to specify rules
(2) Supports both rule-based and learning-based
(3) Utilizes semantic similarities to improve adaptability
(4) Seamless service migration

Framework:
• **CoSMoS**: An abstract component model (models the semantics of components and contexts of users)
• **CoRE**: A middleware to support CoSMoS on various distributed computing technologies
• **SeGSeC**: A mechanism to compose an application by synthesizing its workflow based on the semantics of components and contexts of users.

**CoSMoS** are used by **CoRE** to infer new contexts from other contexts, and also by **SeGSeC** to compose applications based on user’s contexts and rules.
Example 1 cont.

**CoSMoS** (Component Service Model with Semantics)
- Provides abstract *component* model.
- Uses *same semantic* (directed) graph to model:
  - functional, semantic, contextual information and user specified rule.
- Model can be represented in different formats, e.g., in WSDL and RDF.
Example 1 cont.

**CoSMoS** (Component Service Model with Semantics): Examples

- **Microphone**
  - Component: Microphone
  - Operation: recordSound
  - HasType: Component outputSound

- **James**
  - Component: JamesComponent
  - Element: User: James
    - Location: office
    - Value: int 30
    - Activity: meeting

- **Voice recognition service**
  - Component: VoiceRecognizer
  - Operation: convertWav2Str
  - HasType: Binary audio/wav
  - Action: Convert
  - Component: inputSound
    - HasType: Primitive String
  - Component: convertWav2Str
    - HasType: Primitive String
CoRE (Component Runtime Environment)

(1) Discover and execute distributed components
(2) Create and manage pseudo components which represent actual users
(3) Acquire contexts of components and users; and model them in CoSMoS.
SeGSeC (Semantic Graph based Service Composition)

Upon receiving a composition request from a user:
- Discovers necessary components via CoRE.
- Analyzes their information modeled by CoSMoS.
- Composes the requested application by synthesizing its workflow using the discovered components.
Example 2: Goal based
Example 2: Goal based


- Presents a goal-based framework for dynamic service discovery and composition

  Use WSMO for semantic

- An ontology-based conceptual model for describing Semantic web services
- A goal is defined as an objective that a client might have when consulting a web service, and describes aspects related to user desires with respect to the requested functionality
Example 2 cont.

1. Tries to match user goal with the goals defined in the domain ontology.
2. Once a goal is found, platform tries to find tasks defined in the task ontology that can fulfill this goal;
3. With the mapping between services and tasks, platform can identify an available service that implements the task.

Clients can express what they want from services at a higher level of abstraction.
Example 3: Rule based

THE PATIENTS KNOW MORE ABOUT THEIR DISEASES THAN ME. I MUST GET FASTER MODEM, HIGHER SPEED INTERNET ACCESS THAN THEM.
Example 3: Rule based

Typically, an **action** is taken when some **event** happens (e.g., when someone falls)

\[ \text{EVENT} \rightarrow \text{Condition} \rightarrow \text{Action} \]

- Behavior of a care system (i.e., how it reacts to the home and to the user) is specified by policy rules

- Three types of preferences are introduced:
  - **Fixed Preferences** (e.g., heating bath water at 8 pm)
  - **Conditional Preferences** (e.g., switch on light if it is dark)
  - **Resource Preferences** (e.g., display message on TV)

- Value of a variable is substituted when a policy is **executed**, not when it is **defined**.
• Fixed preferences
e.g., heating bath water at 20:00 every day until the required temperature is reached.

```xml
<policy owner="jim@homes.org.uk"
    applies_to="@house5.homes.org.uk"
    id="Prepare bath water" enabled="true"
    changed="2008-03-15T11:12:03">
    <policy_rule>
        <condition>
            <parameter>time</parameter>
            <operator>eq</operator>
            <value>20:00:00</value>
        </condition>
        <action arg1="on" arg2="/water_heater"
                arg5="/temperature=bath_temperature">
            device_out(arg1, arg2,,,-arg5)
        </action>
    </policy_rule>
</policy>
```
Example 3 cont.

• Conditional preferences
  e.g., 1. Prefers a louder TV volume during daily activities (9AM to 11PM).
    2. Wants to keep a lower volume at night.

```xml
<policy_rule>
  <condition>
    <parameter>time</parameter>
    <operator>in</operator>
    <value>09:00:00..23:00:00</value>
  </condition>
  <action arg1="tv_volume" arg2="8">
    set_variable(arg1,arg2)
  </action>
</policy_rule>
```

```xml
<policy_rule>
  <condition>
    <parameter>time</parameter>
    <operator>in</operator>
    <value>23:00:00..09:00:00</value>
  </condition>
  <action arg1="tv_volume" arg2="3">
    set_variable(arg1,arg2)
  </action>
</policy_rule>
```
• Resource preferences
  e.g. Display a message somewhere in the same room

```xml
<policy_rule>
  <trigger arg1="sms_in" arg2="mobile">
    device_in(arg1, arg2)
  </trigger>
  <condition>
    <parameter>time</parameter>
    <operator>in</operator>
    <value>09:00:00..21:00:00</value>
  </condition>
  <action arg1="display"
    arg2="$:nearby_display"
    arg5="content=.parameter_values">
    device_out(arg1, arg2, , , arg5)
  </action>
</policy_rule>
```

```xml
<variable id="nearby_display"
  owner="jim@homes.org.uk"
  applies_to="@house5.homes.org.uk"
  value="func=display,location=:user_place"
  changed="2007-02-23T23:50:00"/>
```
Example 4: AOP
AOP paradigm

- Improves modularity and configurability
- Upon events of a specified type additional functionality can be executed, which has not been the part of the original code

- Programs are extended in a non-intrusive manner
  - No code changed, original program remains the same
- Joinpoints points in a program, where additional functionality can be weaved in
- Aspect = advice (code) + pointcut (points to a location in program where new functionality is to be included)
- Weaver the functionality that includes the functionality specified in the advice in the location specified by pointcut during program execution
Example 4 cont.

✓ BPEL engines
  • interpret BPEL process models
  • publish navigation events to the outside world
✓ BPEL processes can be treated as programs
✓ WSs implement functionalities, can be treated as the functionalities to be weaved in upon a navigation event (advices)

The BPEL’n’Aspects Approach

• Define aspects in terms of WSPolicy
• Allow attaching these aspects to:
  ✓ Processes,
  ✓ Process instances
  ✓ Per activity, link, variable

• Aspect is a combination of:
  any Web Service (advice), a point cut, and advice type (before, after and instead)
Example 4 cont.
Example 4 cont.
Summary
Summary - Back to U-Care

16 February 2010                                Service tailoring approaches

User select his/her goals

Ontology of Goals, and Tasks

Goal based

Semantic (ontology)

Required services are selected from existing services

Configuration of individual services

Composition of configured services

Rule based

BPEL
Summary - Back to U-Care cont.

Context-aware ECA rules

Event happens

Identification of events & conditions

Configuration of individual services

Composition of configured services

Ontology of Contexts and Events

Context-aware

Semantic (ontology)

Rule based Context-aware BPEL
“If we do not change direction, we are likely to end up where we are heading”

Confucius