SEVENTH FRAMEWORK PROGRAMME
NMP-2007-3.1-2
New added-value user-centered products and product services

SERVice Oriented **I**ntelligent **V**alue Adding n**E**twork for Clothing-SMEs embarking in Mass-Customisation

D2.3.2 Integrated Knowledge repositories

Identifier

Project Reference No	**FP7-214455**
Deliverable	**D2.3.2 Integrated Knowledge repositories**
Workpackage	**WP2: Development of Knowledge Infrastructure and Knowledge Management Tools**
Nature	**Prototype**
Dissemination Level	**Public**
Date	**30/07/2010**
Status	**Final**
Editor(s)	**Dimitris Vogiatzis, Dimitris Pierrakos (NCSR D)** **Yannis Leousis, Konstantina Papachristopoulou (ATC)**
Document description	This report presents in brief the prototype of the Integrated Knowledge Repositories with the description of the mechanism developed to manage and store the data, as well as to allow the exchange of information between the developed interfaces, namely the PServer and the SERVIVE Style Advisor (StA).

TABLE OF CONTENTS

1 Introduction 3
2 Knowledge Integration: SERVIVE Fashion Ontology 3
3 System Architecture 3
3.1 Structure of the Knowledge Repository 4
3.2 Extracting Attribute Rules 5
3.3 PServer Stereotype Retrieval 6
3.4 Generating Recommendations 6

LIST OF FIGURES

Figure 1: StA database ERD 4
Figure 2: XML file generated from user data on the StA Wizard which is sent to the PServer 5
Figure 3: XML output from PServer 7
Figure 4: Work Flow diagram 7

LIST OF TABLES

Table 1: PServer's recommendation values 6

LIST OF TERMS AND ABBREVIATIONS

Abbreviation	Definition
ERD	Entity Relationship Diagram
PServer	Personalisation Server
SFO	SERVIVE Fashion Ontology
StA	Style Advisor
XML	Extensible Markup Language

1 Introduction

In the framework of Task 2.1 the domain knowledge sources as well as the gathering of this knowledge which are managed in the project were investigated and specified. Moreover, an integrated analysis of all these sources has been performed and thus the structure of the knowledge types, their representative features and the associations between them have been identified.

In the present report there is a brief presentation of the prototype Integrated Knowledge repositories with the description of the mechanism developed to manage and store the data as well as to allow the exchange of information between the developed interfaces, namely the PServer and the SERVIVE Style Advisor (StA).

2 Knowledge Integration: SERVIVE Fashion Ontology

Knowledge in the fashion domain comes from various sources. Knowledge about garments and the various human styles is usually provided by the manufacturers and fashion experts of the garments, as well as human morphology. Although there are associations between the above types of information source, the majority of knowledge that this information conveys is difficult to be managed efficiently. Thus, there is the requirement for processing this information, extracting the available knowledge and presenting it in a more structured and manageable form.

The **SERVIVE Fashion Ontology (SFO)** provides a structured and unified vocabulary to represent human, fashion and manufacturing concepts. The ontology shares a number of common terms and concepts from the above domains and it is further specialised to cover the needs of each part. The SFO was developed in OWL 2.0 with the aid of the Protege 4.1 ontology editor. The ontology captures the experience of a style advisor, which could be stated in abstract terms as: given some **body measurements** and some **facial features** infer the body type, and subsequently suggest **a garment type**, as well as some **garment colours**.

3 System Architecture

SFO ontology and the PServer have been integrated into a system that is able to provide recommendation functionality for garments. The system, named **SERVIVE Style Advisor** is a domain knowledge - based mechanism, continuously capturing consumer knowledge (preferences, individual customers design ``creativity''), while guiding consumers in the making of their clothes. Style Advisor enables the accumulation and intelligent retrieval of knowledge acquired from prominent field experts, while at the same time the knowledge base is continuously adapted to customer preferences and buying styles, classified according to well defined customer groups (stereotypes).

The Style Advisor consists of the following modules:

1. **The User Interface**, which is responsible for all the communication between the customer and the system.
2. **The Matching Stereotype engine**, which is connected to the PServer and extracts the stereotype of the particular customer.
3. **The Recommendation engine** which generates the recommendations for each customer based on her stereotypes.
4. **The Knowledge Repository** and Reasoner Engine which corresponds to the storage of the domain's semantic information, i.e., the ontology, and offers the inference functionality via the PELLET reasoner.
5. **The Manager Engine**, responsible for the management of the system functionalities and the communication between the system modules.

3.1 Structure of the Knowledge Repository

The database which supports the functionality of the StA (User interface) is built around the CLOTHING ITEM entity and the TYPE, SYBTYPE, POSITION categorization tables. Each CLOTHING ITEM has an overall colour (COLOUR), a brand (BRAND), related files (RELATED_FILE), COMMENTS, voting per user (STARS_VOTE), available sizes (SIZE/CLOTHING_ITEM_SIZE) and occasions that could be used (OCCASION/CLOTHING_ ITEM_OCCASION).The Entity-Relationship diagram of the StA database is presented in the Figure 3 below.

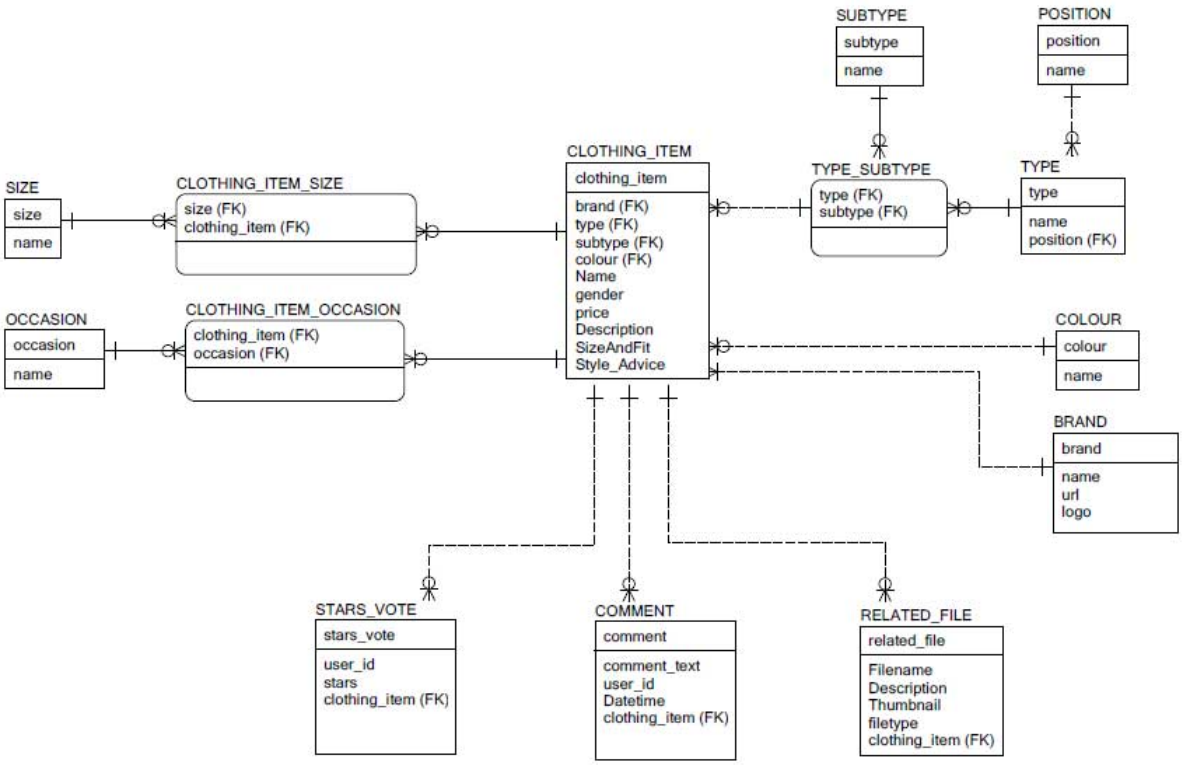

Figure 1: StA database ERD

The user interface communicates the user's data through XML generated files, as presented in the following Figure 3.

In case a registered user has invoked the StA before, his/her attributes are stored in the user profile and can be used every time. In case the user uses the wizard for the first time, then the user attributes are stored to the user profile when the user finishes the wizard.

Any guest user can invoke the wizard anytime and when they finish the wizard they are asked to register, if case they want their attributes to be stored in the community. If the guest declines the registration, then the data are deleted and not stored in the database.

In both use cases, after the finish of the wizard an XML file is built which contains a valid or empty user id (in case of the registered user or the guest accordingly) as well as the user attributes selected in the wizard.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Profile uid="guest or UID">
    <Name>Nansy</Name>
    <Age>32</Age>
    <Sex>2</Sex>
    <HairColor>4</HairColor>
    <HairLength>4</HairLength>
    <EyeColor>2</EyeColor>
    <EyeShape>4</EyeShape>
    <FaceShape>1</FaceShape>
    <SkinTone>5</SkinTone>
    <Nose>2</Nose>
    <Mouth>2</Mouth>
    <BodyShape>5</BodyShape>
    <Height>185</Height>
    <Weight>95</Weight>
    <DressSize>XL</DressSize>
    <Bust>123</Bust>
    <Waist>232</Waist>
    <Hips>1212</Hips>
    <AcrossShoulder>123</AcrossShoulder>
    <BackLength>12</BackLength>
    <LegLength>323</LegLength>
    <ShoeSize>38</ShoeSize>
    <ColourType>2</ColourType>
    <ColourLook>2</ColourLook>
    <StyleType>2</StyleType>
    <ShoeOrder>1,2,3,4,5,6,7,8</ShoeOrder>
    <GiftOrder>1,2,3,4,5,6,7,8</GiftOrder>
    <WomenOrder>3,2,1,4,6,5,7,8</WomenOrder>
</Profile>
```

Figure 2: XML file generated from user data on the StA Wizard which is sent to the PServer

3.2 Extracting Attribute Rules

Using PELLET and the attribute rules, the particular individual is also assigned to an equivalent class. For instance, the individual Mary with the object properties described above, can be assigned to the class that describes her **Style Colour**, which in Mary's case is ``Spring''.

3.3 PServer Stereotype Retrieval

Having inferred the individual's attributes, the PServer's stereotypes are triggered. This step is realized by initially assigning to the PServer's attribute values the corresponding ontology class, as represented by the attribute rule and subsequently retrieving the stereotypes which correspond to the appropriate attribute values. The triggered stereotypes are represented by a set of features which are unique for the particular stereotypes. In the fashion domain that we exploit, the stereotype features describe the various characteristics of the garments, such as color, material, style, etc. Using Mary's case, the class (attribute rule) ``Spring'' is mapped to the attribute ``human color style'', with the attribute value ``Spring'', which triggers the stereotype ``SpringStereotype''. The features of the stereotype are ``jacket.loose=1, jacket.color.azur=1 ...''.

The above process is depicted below:

```
 attr..rule:Spring  ---> attr:Spring
attr:Spring  ---> stereotype:SpringStereotype
 stereotype:SpringStereotype  -->ftr:jacket.loose=1
 ftr:jacket.color.azur=1  ....
```

Apart from the retrieval of a rigid stereotype, i.e., a stereotype that represents a style advice rule defined by a fashion expert, PServer offers the option of the retrieval of a flexible stereotype, i.e., a stereotype that corresponds to user preferences. Similar to the rigid stereotypes, a flexible stereotype is triggered when the appropriate attribute value is specified. In this manner two types of stereotype are selected: a stereotype corresponding to what experts consider appropriate and a stereotype corresponding to what other people consider appropriate.

3.4 Generating Recommendations

Following the PServer analysis of the sent XML with the user attributes, a new XML is created containing all TYPES/SUBSTYPES that fit better, according to the set of rules defined, as seen in Figure 5 below. The values returned for each type/subtype pair are used to signify which pair is appropriate for selection, according the table below.

1	best recommended match
0.5	recommended match
0	neutral
-1	not recommended match

Table 1: PServer's recommendation values

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--?xml-stylesheet type="text/xsl" href="/resp_xsl/singlestereot_profile.xsl"?-->
<result>
  <row><ftr>garment.overall.dress.DropWaist</ftr><val>1</val></row>
  <row><ftr>garment.overall.dress.EmpireLine</ftr><val>1</val></row>
  <row><ftr>garment.overall.dress.Flated</ftr><val>1</val></row>
  <row><ftr>garment.overall.dress.Princess</ftr><val>1</val></row>
  <row><ftr>garment.overall.dress.Shift</ftr><val>1</val></row>
  <row><ftr>garment.overall.dress.ShirtWaist</ftr><val>1</val></row>
  <row><ftr>garment.top.jackets.fitted</ftr><val>1</val></row>
  <row><ftr>garment.top.jackets.Long</ftr><val>0.5</val></row>
  <row><ftr>garment.top.jackets.Straight</ftr><val>0.5</val></row>
  <row><ftr>garment.bottom.skirt.Circular</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Flared</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Gathered</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Gored</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.JeansStyle</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Long</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Pleated</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Short</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Split</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.Tulip</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.WrapOver</ftr><val>0</val></row>
  <row><ftr>garment.bottom.skirt.YokeFlare</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.BootLeg</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.Classic</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.Flared</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.Jeans</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.Jodphur</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.NarrowLeg</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.PegTop</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.Pleated</ftr><val>0</val></row>
  <row><ftr>garment.bottom.trousers.WideLeg</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.Belted</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.ChanelStyle</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.Collarless</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.Hoody</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.JeansJacket</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.Peplum</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.ShawlCollar</ftr><val>0</val></row>
  <row><ftr>garment.top.jackets.MilitaryStyles</ftr><val>-1</val></row>
</result>
```

Figure 3: XML output from PServer

The XML from PServer is then processed and the SERVIVE Community database is queried for all items that match the rule > 0, sorted by best match. The results are then presented as garments in SERVIVE Showroom.

The figure below presents the overall Work Flow.

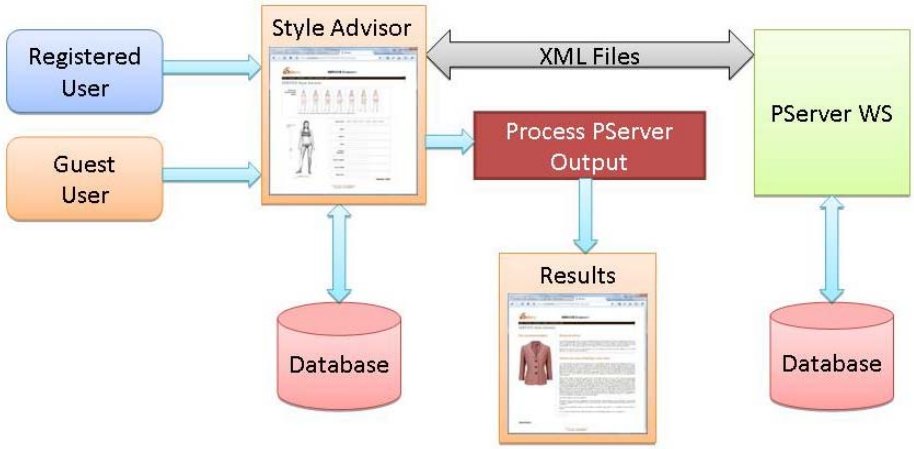

Figure 4: Work Flow diagram