Deliverable D3.2
Shell for Affective User Modelling

Gerd Andersson, SICS       Ana Paiva, INESC-ID
Yasmine Aрафа, ICSTM       Moisés Piedade, INESC-ID
Luис Botelho, ADETTI       Pedro Ramos, ADETTI
Dário Mourао, ADETTI       Phoebe Sengers, GMD

Abstract
This document constitutes Deliverable D3.2 of the IST Project SAFIRA, on the Shell for affective user modelling. It presents first an analysis of the different methods for affective user modelling and mechanisms by which emotions can be ascribed to a user through the analysis of his/her behaviour. In particular we investigate the current state-of-the-art in affective elements that need to be part of the user model and identify the salient components that are necessary for delivering affective modes of interaction between system and user. Then, the document focuses on the specification and first prototype of affective user modelling, which will be integrated in the SAFIRA toolkit and applied in at least one of the demonstrators. In the final part of the deliverable we will also provide a description on how affective user modelling will be used in two of our demonstrators (FantasyA and James the Butler).

Keywords
Emotions, Affective Acquisition, Generic User Modelling, Affective User Modelling.
Deliverable D3.2
Shell for affective user modelling

<table>
<thead>
<tr>
<th>Deliverable Number:</th>
<th>D-SAFIRA-WP3-D3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td></td>
</tr>
<tr>
<td>Contractual Date of Delivery to CEC:</td>
<td>February 28, 2002</td>
</tr>
<tr>
<td>Actual Date of Delivery to CEC:</td>
<td>March 7, 2002</td>
</tr>
<tr>
<td>Editor:</td>
<td>WP3</td>
</tr>
<tr>
<td>Deliverable Type:</td>
<td>P-Public</td>
</tr>
<tr>
<td>Deliverable Nature:</td>
<td>R-Report</td>
</tr>
<tr>
<td>Workpackages Contributing:</td>
<td>WP2, WP3, WP4, WP5, WP6, WP7</td>
</tr>
<tr>
<td>Partners Contributing:</td>
<td>INESC-ID, ADETTI, GMD, SICS</td>
</tr>
</tbody>
</table>

This document is a Deliverable of the SAFIRA project (“Supporting Affective Interactions for Real-time Applications”), a project performed within the IST programme (project identifier IST-1999-11683).

The SAFIRA project is performed by a consortium consisting of the following partner organisations:

INESC-ID – Instituto de Engenharia de Sistemas e Computadores (P, Coordinating Partner)
ADETTI – Associação para o Desenvolvimento das Telecomunicações e Técnicas de Informática (P)
DFKI – Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (D)
FhG – Fraunhofer-Gesellschaft (Previously GMD) (D)
ICSTM – Imperial College of Science, Technology and Medicine (UK)
ÖFAI – Austrian Research Institute for Artificial Intelligence (A)
SICS – Swedish Institute of Computer Science (SE)

Project Manager: Ana Paiva
INESC-ID
Rua Alves Redol 9, 6º Esq.
P-1000-029 Lisboa
Portugal
Tel.: +351 21 3100219
Fax: +351 21 3145843
Email: ana.paiva@inesc.pt

Project Homepage: http://gaips.inesc.pt/safira

Project Email: safira@gaips.inesc.pt
Executive Summary

This Deliverable (D-SAFIRA-WP3-D3.2) is the second deliverable from WP3 (Affective Input) of the IST-sponsored SAFIRA project. It contains a detailed description of the research and technological analysis made in the first six months of this workpackage (October, 2000 to March, 2001) on the area of user modelling and affective interactions. Based on some of the state of the art findings a specification for the SAFIRA component for affective user modelling was produced and is now here reported. This document reports on the first development done in the area of affective user modelling within the SAFIRA project, describing in detail its aims, architecture, basic elements, planned integration in the Safira toolkit and subsequent use in the demonstrators.

The research and results are primarily based on collaborative work between the partners of this workpackage as well as all consortium members.

The objectives of WP3, as per the Technical Annex, p 36, were the following:

1. Classify and test different types of input that will capture human affect.

2. Specify and construct a component that will analyse user’s behaviour with affective objects, aiming towards its integration in the toolkit.

3. Extend current user modelling techniques to represent affect and personality of the user. Specify and develop a shell for affective user modelling.

This deliverable focuses on the last objective. It will include a small overview of the area of affective user modelling and then the description of one components for the acquisition of affective signals from the user and its associated representation.

On the whole, this work contributes substantially to the overall SAFIRA project objectives, in providing innovative components which not only embed aspects of affective computing and affective interactions, but also extends this area into the challenges of this decade with the advent of personalisation and adaptation to the users.

The research and development has been completed in a timely manner and of a uniformly high quality. It has generated significant and substantial research directions. One paper has already been presented at the 2nd Workshop on Attitude, Personality and Emotions in User-Adapted Interaction and one was published in the book “Affective Interactions” by Springer.

The following chapters are organised to reflect the work done in WP3 and thus are divided as follows:
Chapter 1 acts as a general introduction expanding on the reasoning and requirements for affective user modelling in general. It can be read as a survey in the intersection of the areas of affective interactions and user modelling.

Chapter 2 provides an introduction to affective user models in Safira, and describes the development description of the affective user modelling shell.

Chapter 3 provides the description of the application of the mechanisms just described in the demonstrators we are about to construct.

Chapter 4 gives some preliminary conclusions.
# Table of Contents

1. Introduction to Affective User Modelling in Safira .................................................. 9
   1.1 From Personalisation to Affective User Modelling .............................................. 10
   1.2 Affective User Modelling in Safira .................................................................... 12
   1.3 Generic User Modelling ...................................................................................... 13

2. General Framework for Affective User Modelling .................................................. 14
   2.1 Representation of Affective User Models in Safira .............................................. 17
      2.1.1 Emotions in the AUM ............................................................................... 17
      2.1.2 Emotional Profile .................................................................................... 21
      2.1.3 Goals ....................................................................................................... 22
      2.1.4 Knowledge ............................................................................................. 25
   2.2 The Architecture of the Affective User Modelling Component .............................. 25
      2.2.1 Inference Engine ..................................................................................... 26

3. Use of Affective User Models in Safira ................................................................... 32
   3.1 Integration in FantasyA ..................................................................................... 32
      3.1.1 Use of Affective User Models in FantasyA using SenToy .............................. 33
      3.1.2 How the component works ......................................................................... 34
      3.1.3 Using the AUM Component ...................................................................... 38
      3.1.4 Defining Events, Agents and Objects in the AUM ....................................... 40
      3.1.5 Overall architecture .................................................................................. 42
   3.2 Affective User Models for James the Butler ......................................................... 44
      3.2.1 Context Sensitive Personalisation .............................................................. 45
      3.2.2 User Profile ............................................................................................. 49

4. Conclusions .............................................................................................................. 57

Bibliography .............................................................................................................. 58
1. Introduction to Affective User Modelling in Safira

In this section we review some implementations of current affective user modelling techniques. This work was performed in Task 3.2. We describe the scientific background and theories of affective user modelling and emphasize our contribution to this area of research.

As per the technical annex:

“User modelling aims at capturing information about the user and infer his/her preferences, goals, plans and knowledge. However, from the observation of user’s actions and behaviour and given the strong link between affect and cognition, it is now more and more important for any application to model and reason about the user’s affective state. The data captured through the keyboard, affective wearables or affective objects can be used for the construction of what we call an “affective user model.”

In particular, the main objective that will be handled in this deliverable is:

“To extend current user modelling techniques to represent affect and personality of the user. Specify and develop a shell for affective user modelling.”

To address this issue, we will briefly analyse general user modelling techniques, in one hand and on the other, affective user modelling. The research here documented will help to establish the generic affective user modelling system that we built for the Safira toolkit.
1.1 From Personalisation to Affective User Modelling

It has become increasingly apparent in the fields of Intelligent Agents and Human Computer Interaction, as we attempt to construct systems that are adaptable to humans, that modeling the users is an essential part of those systems.

It is necessary for the system to have knowledge about the goals, characteristics, states of knowledge, and motivations of the users, in order to adequately to respond them and hence personalise the interaction established. So, user modelling is a fundamental mechanism to achieve such personalisation in the communication between interactive computer systems and humans. It is expected that interactive systems provide facilities and feedback that are responsive and adaptive to the users' needs, knowledge, goals, desires or characteristics.

In recent years, new and more complex interfaces started being built leading to an enormous evolution in the systems' usability. This evolution fostered the user and learner modeling areas since it become apparent that adaptability to the individual is an important issue in usability, and user modelling constitutes the basic mechanisms to achieve such individualisation.

Imagine a situation where a novice user is operating a system which will guide him to get the appropriate information from a world wide distributed data-base. The information the user is looking for is about hotels in China and about hotel reservations. For the system to be able to filter information, give adequate advice, guide the user to adequate information, a model of the user's wants and profile needs to be known by the system (the user may be a student looking for cheap hotels, or may be a tourism operator who wants to evaluate capacities of hotels in China).

Many other illustrative situations can be given where the use of user models is a needed fact in the individualisation. Its power can range between just simple adaptability to highly user adaptative dependent systems.
But user modelling is not limited to being used for adaptability. It can be used for other purposes, such as the simulation of users for avatars, learners simulation for teacher training, or even, for the creation of control models of synthetic characters.

Areas of application range from natural language interpretation, multi-modal communication, product customisation, intelligent learning, information filtering, etc.

In general, the question of adaptation to the user has been approached with a focus on acquiring and representing the user's preferences, goals, historical or state of knowledge. Such elements compose a user model and are normally inferred through the use of diverse types of techniques, such as machine learning techniques to capture users' preferences, or cognitive diagnosis techniques to capture users' misconceptions in a domain knowledge [Self 92].

More recently, and influenced by the emerging field of Affective Computing, i.e. "computing that relates to, arises from or deliberately influences emotions" [Picard 97], a new area called **Affective User Modelling** (AUM) has appeared and is now giving its first steps. There, the goal is to capture emotional aspects of the user, being them obtained through devices that obtain physiological signals, through the user’s facial expression, or even by the behaviour of the user in certain situations. In this context Elliot and all. in [Elliott 99] defined Affective User Modelling (AUM) as the capability for a system to model the user's affective states. Indeed, as Humans interact directly with computers, it is critical for this interaction to be empowered with affective components, not only in the expression of affect by the system but also in the capture and reaction to the affective states of the user.

A substantial amount of work has already been done in the acquisition of a user's affective states based on physiological signals (e.g. Affective Computing group at the MIT Media Lab). Most of the research development is based on multi-modal forms of input as affective, either using what is named as **affective wearables** [Picard 97], or by using other forms of input such as speech recognition [Ball 99] or facial expression recognition [Wehrle 00]. This type of affective recognition has been surveyed in D3.1.
It is a fact that physiological aspects of emotional states can be identified and collected through the use of different types of sensors. However, nowadays the typical interactions established with computers applications still use traditional interfaces, as keyboard and mouse input. But even with such limited type of input it is possible to capture certain emotional states of the user. Indeed, while interacting with computer applications, users communicate their affective states not only through body expressions and physiological signals, but also through their behaviour. In a well defined context, as a game, a tutoring situation, or a selling situation, the users' behavioural responses may be a path to predict, recognise and interpret the user’s affective states.

1.2 Affective User Modelling in Safira

In Safira, we will explore two types of situations. The first one, where multi-modal input is being built with the SenToy and the Influencing Machine, capturing emotions of the user is based on the types of physical actions performed by the user. The second one, in the application of James the Butler, the capture is based on the utterances and actions performed by the user through the keyboard.

To obtain an umbrella in which to include these two approaches to affective acquisition, we propose in this deliverable not only to use the input based on the two tangible interfaces, but also to use the situations the user is faced with, as well as the user's behaviour. We will ascribe affective states to that user using an approach named Cognitive-Based Affective User Modeling (CB-AUM). This approach has already been briefly proposed by Elliott et al. [Elliott 99] and extended in [Martinho 00]. Its main idea is that user’s affective state is influenced by the situation the user is in. So, to ascribe some affective states we will have a model of the situation (context) which will be used for the inference on a “possible” state of the user. Note that this approach complements the work reported in Deliverable 3.1 as it can be used in combination with such affective physical interfaces.
1.3 Generic User Modelling

The construction of user and learner modelling systems has some repetitive tasks, and some procedures and techniques can work along a vast range of domains and situations.

So, one of the problems of the user and learner modelling community is to abstract the techniques and mechanisms found in several modelling systems and use them in the future in an independent manner for different domains and applications. That's the problem of generality and independence in user and learner modelling.

The proposed approach to user modelling combines that generality with the affective aspects in which the capture will be made, proposing a generic framework for affective user modelling. This is done by taking an approach whereby affective user modelling is seen as an independent task performed by an independent component in the Safira toolkit.

To do that, we got inspiration on the development of generic user modelling systems, in particular user modelling shells. Tracking back such systems, lead us to the work by T. Finin published in 1986, where his “General User Modeling System (GUMS) [Finin 86] was described. This was the first attempt to generalise the process of acquiring, reasoning and using user models. Other systems were developed such as UMT [Brajnik and Tasso 94], BGP-MS [Kobsa and Pohl 95], DOPPELGÄNGER [Orwant 95], TAGUS [Paiva and Self 95] and um [Kay 95] all addressing the generalisation of user modelling and the construction of tools that allow the creation of user models within a particular application.

Here we will try to use the ideas that are representative of user modelling shells and extending them to cover the affective modelling as a generic task. This generality is essential for the integration of this component in the Safira toolkit.
2. General Framework for Affective User Modelling

Since the seventies, both the learner and user modeling research communities have dwelt into the problem of acquiring, representing and using users' characteristics and attitudes to personalise the interaction. From a focus on how to infer students' states of knowledge and diagnose their misconceptions, to how to obtain the users' preferences, many systems have been build and used (see [Kobsa 89] for a good collection, or more recently [UM 99] and [UM 2001]). Moreover, and to deal with different types of demands, UM shells were built as a way to facilitate applications to integrate UM functionality (for example and among others: BGP-MS [Kobsa and Pohl 95], UM-toolkit [Kay 95], TAGUS [Paiva 95]). Most of these UM systems and shells allow the user model to contain different types of information such as: knowledge, beliefs, characteristics, and preferences.

In general, the different information types found in traditional user models are:

- **Characteristics**: normally captured within a profile of the user, such as age, gender, and interests. This information can be relevant if stereotypes are used to infer other more specific information, such as preferences.

- **Domain Knowledge**: the beliefs of the user about a domain of knowledge of the application. This type of content is relevant for Tutoring Systems or advisory systems and aims at responding to the user’s state of knowledge and providing with adequate advice.

- **Preferences and Needs**: normally used in interactive systems to advice the user, or in interface agents to select the most appropriate information to provide. Most of the information retrieval support agents are based on the representation of the information “needs” or “preferences” of the users (among others see [Billrus]).

- **Goals and Plans**: used in problem solving or task execution situations, both on learning environments and help systems. The need for goals and plans is an
immediate result of the need to support the users to achieve *adequately* their tasks. A large and important amount of work has been done in the area of acquiring the users’ plans (e.g. [Kautz]).

Most of these elements can be obtained, with some degree of certainty, by the analysis of the user’s behaviour. To express this uncertainty of the models, numerical factors and probabilistic reasoning are also used.

But, in order to give a computer system the ability to act in an affective consistent manner with the user’s emotional state, it is necessary to build it the means to accurately acquire and reason about such characteristic of the user. Such systems must have what is called the Affective User Model (AUM). Thus, affective user modelling involves to represent, acquire, and hypothesize about the user's emotional states. Hence, we need:

1. to include a representation of the emotions or affective states in the user model;
2. to develop techniques to make predictions and inferences around these emotions;
3. to know when to use this data according to the situation the user is in.

So, the main questions to address are:

- How to acquire the user's affective states?
- How to represent them?
- How to use these representations effectively?

To begin with, we will discuss what can be regarded as the bare-bones of the whole framework for affective user modelling, a kind of skeleton that we will augment with affective components in the user modelling scene.

The key point introduced here is that the affective user modelling system is viewed as an independent component, and the user modelling as a whole process on its own, therefore not being a part of an application. Its embedment in the application is based on the integration procedures that are being developed within workpackage 2.
Obviously, this approach has many arguments against it. The main criticism will certainly be that one cannot regard user modelling independently from the system that uses it because it relies too strongly on the domain area of the application. However, although such criticism is valid, one can assume that the communication of the two systems/components is good enough to overcome that dependency problem. That will be one of the major assumptions of this framework (and indeed the whole idea of the components in the Safira Toolkit).

So, based on this key point, there are three main principles about affective user modelling in the context of Safira which the whole framework stands upon. They are:

- Affective user modelling is a task/process to be performed by a component that aims at creating and maintaining a model of affective elements of individual users;

- Affective user modelling is a task on its own, independent from other higher level tasks, which might combine a user modelling component.

- Affective user modelling assumes that there are explicit symbolic assumptions representing some affective state of the user. Such assumptions can be used by the system (or other component) to establish an affective interaction with the user.

Having drawn an idea of the framework, it is possible now to deepen some of the particularities of each of its elements.

We'll start with the user models themselves. Although the emphasis will be placed on how the process of acquiring them is defined, it is important to note that such an operation that takes as an argument a represented affective user model and the result will be another affective user model.

We will consider in Safira that an AUM is a compound of elementary pieces of information (acquired and believed by the component). These are the building blocks of the Affective user modelling process and they constitute the smallest pieces of
information that the component is able to identify about the affective state of the user. As main restriction, we will only consider that among such elementary pieces of information the system is able to ascribe emotions to the user.

2.1 Representation of Affective User Models in Safira

The first step considered in the development of an affective user model, consists in defining a discrete set of emotional states the system can infer, to describe the user’s current affective state. Once the set of emotions an user can have is settled, the next step consists in determining the pre-conditions of those emotional states, which must be true so that the associated emotion can be experienced, i.e. one must determine the eliciting situations that give rise a certain emotion. This issue is addressed in the section referring the inference engine.

The third aspect, and probably the most difficult to achieve, that one must have in consideration in order to build an affective user model is the dynamics of the emotional system. The kernel issue of this aspect, is to know how long does an emotion last in the system, i.e., how long can the effect of previous inferred emotions be relevant for determining the user’s current emotional state, and how much does it bias this evaluation according to how recent is the emotion.

To support this process of building the Affective User Models for each application, we have defined a set of generic elements that will be part of such Affective User Model. These elements are: user emotions; user goals; user knowledge and user profile (which includes the emotional profile). The next sections will describe the attributes for each of these elements that will compose an AUM in Safira.

2.1.1 Emotions in the AUM

Since the framework for affective user modelling is based around the concept of emotion (we could have instead looked at mood or personality but as justified in
previous deliverables we decided to concentrate our modelling in the emotional states), it is necessary to provide a definition for an emotion in the system.

Table 2.1 shows the specification for the attributes for an emotion.

We will be ascribing emotions to the user based on the user's behaviour and on the events of the world. Hence, we need a cognitive theory of emotions that consider and work with such stimuli. From the vast array of theories of emotions stating how emotions are generated, and considering the surveys done in WP2 and WP4, we chose the OCC theory [OCC] as the support for the construction of our affective user model. In the remaining of this section, we will describe how the OCC theory can be used for the representation of the user's affective states.

Based on the OCC theory, the affective elements of a user affective model can be characterised by two type of data: the user emotional profile (which will be explained later) and the emotions themselves. Concerning the emotions themselves, the attributes considered for the description of an emotion, are according with the OCC theory of emotions.

Table 2.1 - Emotion Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>The id of the emotion class being experienced</td>
</tr>
<tr>
<td>Valence</td>
<td>Denotes the basic types of emotional response. Neutral, positive or negative value of the reaction.</td>
</tr>
<tr>
<td>Subject</td>
<td>The id of the agent experiencing the emotion</td>
</tr>
<tr>
<td>Target</td>
<td>The id of the event/agent/object towards the emotion is directed</td>
</tr>
<tr>
<td>Intensity</td>
<td>The intensity of the emotion. A logarithmic scale between 0-10</td>
</tr>
<tr>
<td>Time-stamp</td>
<td>The moment in time when the emotion was felt</td>
</tr>
</tbody>
</table>
The component that originated the emotion: the physical component or the AUM component

The attribute *Class* describing an emotion refers to the type of that emotion. According to the OCC theory, the emotions are organized hierarchically. This hierarchy is in sum defined by three types of reactions, depending on three types of aspects of the real world: events, agents and objects. These aspects are responsible for causing emotional reactions. Thus, the three main branches are:

- **Event-based emotions**: pleased or displeased reactions to events.
- **Attribution emotions**: approving or disapproving reactions to agents.
- **Attraction emotions**: liking or disliking reactions to objects.

The OCC structure of emotions defines, in this way, a hierarchical organization for emotion types. An emotion type represents a family of related emotions differing in terms of their intensity and manifestation, i.e., each emotion type can be realized in a variety of related forms e.g. fear with varying degrees of intensity – *concern, fright, petrified*.

The attribute *Valence* describes the value (positive or negative) for the reaction that originated the emotion. According to this theory, emotions are always a result of positive or negative reactions to events, agents or objects.

The *Subject* and *Target* attributes for emotions, define the entities related to them. The *Subject* defines the agent experiencing the emotion and the *Target* defines the event, agent or action that originated the emotion.

One must have associated to every emotion, an attribute of *Intensity*, which is assigned with different values depending of the different situations that gave arise to that particular emotion. The value for the intensity parameter is calculated from the difference between the emotion’s activation minimal threshold (*Activation Threshold*)
and the intensity potential value of the associated emotion directly evaluated from the eliciting situation (Emotion Potential). Thus, the formula for the intensity parameter for a specific emotion ($em$) when it is generated, is as follows:

\[
\text{Intensity}_{em} = \text{Emotion Potential}_{em} - \text{Activation Threshold}_{em}
\]

The value resulting from this formula represents the intensity of an emotion, when the emotion is created. However, the intensity of an emotion does not remain constant during its life cycle in the system. Since the moment it is generated, until the moment it is deleted from the system’s emotion repository, the intensity of an emotion must be attenuated through time in order to reflect the dynamics of the emotional system itself. This characteristic reflects the notion that an emotion does not last forever and does not affect the evaluation of the subsequent emotional states in the same way. According to this concept, Picard proposes a decay function for emotions [Picard 97], which characterizes intensity as a function of time. At any time ($t$), the value for the intensity of an emotion ($em$) is given by the formula:

\[
\text{Intensity}_{em,t} = \text{Intensity}_{em,t_0} \times e^{-bt}
\]

The constant value $b$ determines how fast the intensity will decrease over time. This value can be controlled in order to reflect the short or long duration of the emotions. The value $\text{Intensity}_{em,t_0}$ refers to the value of the intensity parameter of the emotion ($em$) when it was generated. This value is referred in the formula (1) above as $\text{Intensity}_{em}$. When the value of $\text{Intensity}_{em,t}$ reaches zero, the emotion ($em$) must be removed from the system’s repository, meaning that that specific emotion will no longer be part of the agent’s emotional database. This condition will be verified in each cycle of the system for the system’s update. Note that these values are dependent on the emotional profile of the user.

The Time-stamp attribute corresponds to the system time in which the emotion was generated. This value is used to compute the elapsed time since the emotion’s creation until the present time of the system. This elapsed time is then used in the time scale for the formula (2) above.
Finally, the *Origin* attribute defines the component that created the emotion. Since we have two components that can originate emotions (the physical component and the affective user modelling component), it is necessary to distinguish between emotions that were created by each one because the virtual environment may react differently for an emotion coming from the physical component and an emotion coming from the AUM component.

Although represented by the attributes just provided, one must note that in Safira, emotions stored by the AUM system are of two types:

- Potential emotions: represent which particular classes of emotions the current situation is likely to provoke according to the users’ inferred attitudes, goals, and standards of behaviors. Additionally, they state how strong those potential emotions are (i.e. their potential).
- Active emotions: whenever the potential of an emotion overcomes the assumed resistance of the user towards this class of emotion, the emotion is said to be *active*, and the user is assumed to be experiencing it. The user model stores these emotions, along with their intensity, that is, the distance between the user's assumed threshold and the potential of the respective potential emotions. The greater this distance, the higher the intensity of the experienced emotion will be assumed to be.

### 2.1.2 Emotional Profile

Note that the intensity of the emotional states will be calculated based on The emotional profile states the assumed emotional conditioning of the user and is constituted by:

- Emotional class thresholds: represent the assumed emotional “resistance” towards the different classes of emotions based on the OCC structure of the emotion classes. For example, emotion thresholds would model, among other things, describe how
easily the player of the game or the user of the Web-based Wine store could be disappointed.

- Emotional class decays: represent how long the emotions assumed to be experienced by the user last. Although, these values are very difficult to predict during the interaction, they are strongly related with the application.

Finally, as mentioned before, besides emotions, the user’s model must store information about the user’s goals and knowledge, which underlie his behaviour. These concepts must be defined, so that they can be represented in the system.

### 2.1.3 Goals

Goals represent a list of actions the agent must achieve in the future. In this case, this list of actions is initially introduced in a file and it represents the agent’s initial goals.

At this point, it is necessary to distinguish the agent’s goals and the user’s goals. If, for instance, the running application is a game-playing environment, the user’s goal, is simply to win the game, but in order to win this game the user must take the goals of the agent he is controlling as his. The agent’s goals in this example are the goals established in the game context for an agent, which must be satisfied so that the agent can successfully get to the end of the game. Thus, the user’s goals are in particular the agent’s goals established in the context.

The table below describes the attributes that define a goal in the system.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Action</em></td>
<td>The action that the user wants to achieve</td>
</tr>
<tr>
<td><em>Target</em></td>
<td>To what the action is referring</td>
</tr>
</tbody>
</table>
Whom | To whom the action is referring
---|---
Priority | Value in range [1,5] Determines the order of the goals in the list.

During run-time, the system will update this list, so that short-term goals needed to be satisfied at an intermediate stage, can be inserted. The order of the goals is set according to the priority value each one has attached. Goals with higher priority will be evaluated first. As the goals are realized, they are erased from this list.

Each goal has three sub-lists for sub-goals. A list of sub-goals that increase the probability of achieving the main goal – the *Facilitative Goals*; and a list of sub-goals that decrease that probability – the *Inhibitory Goals*. These sub-goals, facilitative and inhibitory, have the same attributes as the goals described in Table 2. Besides these attributes, there is another one describing the degree that each sub-goal facilitates or inhibits the main goal. This parameter is called *factor* and its value ranges from 0 to 1 ([0,1]).

The third list of sub-goals describes the *Necessary Goals*: the sub-goals necessary for the main goal completion. Here is an example of the content of a file describing the goals in a particular game-playing situation.

**File with a possible description of goals:**
In this example (for example associated with a game), the main goal is to find the blue rock, which is in door1 in the floor1. To open that door it is necessary to get the key for it, the key1. One of the goals of the game is to exchange artefacts for money in order to
buy other artefacts. This goal is inhibitory in the sense that the agent/player can give away the key1.

2.1.4 Knowledge

The knowledge of the user is represented by a list of facts describing the objects, agents or events, which were observed in a given situation (for example in a game situation); and the corresponding emotional reactions to them.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type of the fact. It can be a reaction or a description</td>
</tr>
<tr>
<td>Target</td>
<td>The agent, event or object that is referred in the fact</td>
</tr>
<tr>
<td>Time-Stamp</td>
<td>The time when the fact occurred</td>
</tr>
</tbody>
</table>

The current goals and knowledge, as well as the emotions and perceptions, represent the user’s current state. These user’s state attributes will be used in conjunction to infer the next emotional state.

This process is explained in more detail in the next section describing the overall architecture of the affective user model component.

2.2 The Architecture of the Affective User Modelling Component

The Affective User Modelling Component (AUMC), developed in Safira, provides the ability to store and make inferences about the user’s affective state using the OCC theory of emotion as the bases for representing and inferring such state. The Figure 2.1 below shows the architecture of the affective user modelling.
The sub-sections below describe the modules that compose this component and the interaction existent between them.

### 2.2.1 Inference Engine

The inference motor is responsible for the kernel process in the affective user model component. This process receives information about the user’s actual perceptions, his goals, knowledge and the physical emotions / actions corresponding to the emotions or actions directly inferred from the physical interface component. The inference motor will then infer the new current emotional state for the user, given the knowledge of such information.

This module is composed by an appraisal structure that follows a rule-based approach. It has a set of if-then rules that will activate an emotion (the consequent) when a given situation (the antecedent) is verified. This appraisal can have for instance, the rule: if (agent is pleased about an event) then (agent feels happy). In this case, the appraisal must also have the rules that determine when the agent will be pleased with a given...
situation, for instance, if he sees a blue object, he likes blue and one of its goals is to pick the objects he likes.

The information produced by the Inference Module will be used to the updating process to update the user state model. The new inferred emotion will flow directly to the actuators.

2.2.1.1 The Appraisal Structure

The appraisal structure is responsible for the inference in Inference Motor module. This module was built for general-purpose applications and can be seen as an independent component.

The appraisal component is structured for this particular application, in six main modules, one for each emotion. Each module has a JESS rules file associated responsible for producing the respective emotion signal.

This component uses sockets to communicate with the other components that subscribed to it. Information received from provider components (as a consequence of a information subscription) is stored in an internal mailbox. The emotion signals are stored in another mailbox for output and then delivered to components that subscribed them.

Modules can subscribe emotion signals from other modules. This feature enables the specification of rules for an emotion in terms of other emotions, i.e. if the module (e.g. joy) subscribes another module (e.g. sad) we can put a rule in the joy-module that has sad as one of the antecedents (e.g. if (and (sad intensity > 0.8) (agent is pleased about the current event)) then (create joy intensity 0.2)).

The figure below shows the general structure of the appraisal component.
For this particular purpose, the Provider and Client components are in fact the same application that is running in the system. This application must have two different channels for the communication with the AUMC. The application’s output channel sends data to the appraisal and the input channel receives data from it, which will be used in the context of the application. The treatment that this application gives to the data provided by the AUM component is responsibility of the application itself.

2.2.1.2 Filtering

The application the user is running will feed the system with some necessary aspects that must be considered for user’s emotional state assessment. These aspects constitute the synthetic agent’s perceptions of the application’s virtual world. The perceptions the AUMC has about the virtual world are coming from what we call the virtual sensors. Some information passed to the affective model, from these virtual sensors, may not be relevant, at a particular instant, for the inference process. Thus, the filtering process is responsible for, according with the current situation (namely goals and knowledge), determining the relevant information about the virtual world for the inference process. The filtering mechanism is necessary in such a system, in order to increase its
performance. It is important that the system does not overload the inference process with information that is not important for the realization of the task.

The perceptions contain the description of the following attributes.

Table 2.4 - Perceptions

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type of the perception: event, agent, object</td>
</tr>
<tr>
<td>Description</td>
<td>Contains the features of the event, agent or object</td>
</tr>
<tr>
<td>Time-Stamp</td>
<td>The time when the perception occurred</td>
</tr>
</tbody>
</table>

The Filtering Mechanism

The filter module receives two distinct types of data from the virtual world application running in the system. This application will provide information for control and information about the perceptions of the agent in the virtual environment.

The information for control consists on messages referring to the addition or removal of goals, although the treatment of any other kind of control is easily integrated in the AUM component.

Obviously, the concrete set of control messages is dependent of the specific virtual application.

The restriction parameter:

Whenever a new goal is added to the AUM component, if the restriction parameter is set accordingly, this component may automatically filter the set of the future perceptions, according to the goal added by the application.

For instance, if the control message is as shown in the following example, for the addition of a goal in a specific context, and the parameter restriction is set to the
particular feature “colour” with the value “blue”, this means that the AUM component, more specifically, the filter module, will start to restrict the perceptions only to the blue objects.

If no restriction is wanted, the value for this parameter must be set to “none”.

Example of a control message:

```
<message>
  <message-id>2</message-id>
  <control>
    <add-goal>
      <action>get</action>
      <target>key1</target>
      <priority>5</priority>
      <restriction>
        <type>object</type>
        <feature>colour</feature>
        <value>blue</value>
      </restriction>
    </add-goal>
    <time-stamp>17:20:29</time-stamp>
  </control>
</message>
```

The filtering mechanism was developed using a JESS file associated the filter module. This file contains the set of rules established from the restrictions added in the context of the control messages.

For instance, in the above example, the referred control message would generate the following rule:

```
(defrule filter-rule-objects1

  (object (name ?n) (feature-value blue)

    (feature colour) (msg-id ?mid) (time-stamp ?ts))

=>

  (send (compose-output-perception-object ?n colour blue ?mid ?ts)))
```
The ‘compose-output-perception-object’ function is a function that prepares the message to be sent to the appraisal mechanism, according the internal protocol established between the appraisal and the filter mechanisms.

All the perceptions provided by the application, referring to the perceptions of a specific agent inhabiting the virtual world application, which are received by the AUM component or more precisely by the filter module, are internally translated to a fact. This fact is then used in the Jess context. The addition of this fact in the JESS mechanism represents the occurrence of a message of the same type containing the corresponding characteristics.

The Jess engine will then look for a pattern matching between the left-hand-sides of the existing rules and the existing facts in the same system.

When Jess finds the matching fact for the rule, the rule will then be fired (or activated). When the rule is activated, the Jess system executes the function in the right-hand-side of the rule. In this case, when the system finds a fact in the knowledge base matching to the pattern in left-hand-side of the rule above, the system will execute the send function, which transmits the fact to the central processing unit in the AUM component. If the Jess system does not find any fact matching the antecedent of the rule, the rule will not fire and the perceptions received are not sent.

In sum, the application can send control information according to goals to the AUM component so that this component filters the information received from a specific agent of that application in form of perceptions.
3. Use of Affective User Models in Safira

The above framework and component is being used within Safira in two main contexts: the SenToy allied with a Virtual Environment and in the personalisation for James the Butler. We will describe next these two applications, although more detailed and implemented descriptions will be found in deliverables 6.1 and 6.2.

3.1 Integration in FantasyA

In here we will provide only some idea on how the integration is FantasyA is being done. Note that FantasyA is still under construction, we don’t have yet any results concerning its integration. However the ideas are as follows:

*FantasyA is a world guided by the Dreams of seven mighty Gods who, although imprisoned in the godly stones, the KemY’s, use the Dreams to impose their will over the inhabitants of the Land*The user plays the role of an Apprentice Magus of one of the Seven Clans of FantasyA. He/she has finished his training in AlkemHYe and passed the Initiation Ceremony. He/she is ready to leave the Covenant and look for the IdenjaYeh for further details on his quest. In game terms, after an initial background explanation of the world and the clans, the user must chose a clan.

Then, after assisting to its training, where the laws of YertamenH are explained, and after watching his first transformation in the Initiation Ceremony, the Apprentice leaves the safety of the Covenant to look for the IdenjaYeh. Soon, he/she learns that the IdenjaYeh is beyond one of the Portals of the Island. But which one?

The user chooses one and the game starts.
Figure 3.1 - Ronin searching for magic gems in FantasyA

The game will develop around the idea of magic gems that provide the user magic powers to control his/her emotions. The user must find a way out of this magical maze and meet (her) his Clan leader as soon as possible. In his quest the user will find some other apprentices (autonomous characters) with whom the user must duel with. In these duels the most important thing is the magic gems the user captured and his/her control of the emotional states (done though the SenToy) interface.

Based on these situations provided by the game and the SenToy interface, the user modelling task for FantasyA is being build based on the situations, goals of the user and emotions he/her will try to express using the SenToy.

3.1.1 Use of Affective User Models in FantasyA using SenToy

In order to implement the affective user modelling component we have used the Java programming language version 3.1.1 and a specific package for expert systems, the Java Expert System Shell, or for short (JESS) version 5.1. The choice for working with the Java development language was mainly due its portability between the two most used operating systems: Windows and Unix.
3.1.2 How the component works

For better understand how to use the component in the context of an application, it is necessary to understand first, how the system works. The figure below identifies the main elements of the system and shows how they interact.

![Figure 3.2 - The interaction between the components of the system](image)

The main component of this system is the Affective Input Component (see also D3.1), which processes data coming from both the physical component and from the Virtual
World component (more specifically from the Filter module). The execution of these two processes is done in parallel, using threads corresponding to each one of them. As an aside, the execution of these threads is not completely parallel because we are not using a multi-processor platform. We then must see it as a pseudo-parallel mechanism.

The Affective Input Component when started performs a series of initialising processes in order to load the initial definitions. For instance, it loads the settings file and the reserves the ports and addresses in memory for the communication between the components, which is accomplished via sockets using the TCP/IP protocol. The messages sent between these components follow a XML structure, which definitions can be found on the descriptions of the components involved. This module then enters in a waiting stage, waiting for the registration of the two components that must be attached to it. When these components register themselves in the Affective Input Component, each of the two threads performs an infinite cycle that keeps receiving data from the components they correspond to, and transforming this data to an appropriate format so it can be used by the expert system mechanism. The next sections explain the relevant processing characteristics of each one of the threads.

3.1.2.1 Processing the physical data

The information provided by the sensors in the plush-toy conveys the gesture that was actually performed, however this information is at this stage, simply raw data. So, it is firstly submitted to a signal processing module, which interprets the signals produced and controls the accuracy of the data. This process then produces the information of the gesture that was interpreted along with its characteristics like intensity, acceleration and time stamp. The resulting information describing the gestures and their parameters are then processed by the thread responsible for the physical data processing in the Affective Input Component.
The processing of the physical data consists firstly on the translation of the information of the XML structured message, like the one shown in the section about *Using the Physical Interface Component*.

The XML message is parsed into an adequate syntax, which can then be used in the context of the expert system shell (the JESS package) and which is composed only by facts and rules. Therefore, the XML message is translated to a fact or to various related facts that hold the equivalent information. The rules for the JESS module in this component correspond to the mapping between gestures and actions or between gestures and emotions. These rules can be edited in the JESS file associated to this module.

Once these facts are dynamically introduced in the JESS module, the expert system then performs a *run*, that is, it checks all the existing rules to verify if there is any rule that must be fired, or activated. The rules are activated when their antecedent, which can be composed by one or more facts and by prepositional expressions, is true, i.e. when all the facts of the antecedent match with existing facts in the JESS list of facts and all the expressions of the antecedent are evaluated true.

When one or more rules are activated, the JESS module executes the corresponding consequents, which are functions. These functions are used, once they are activated, to re-translate the produced fact in the corresponding XML message. This message is then sent to the components the Affective Input Component is attached to. In the Figure showing the overall functioning of the Emotional Input system, the Affective Input Component is registered in the Virtual World Component, which receives the generated messages, so that the information produced corresponding to an emotion or action can be virtually represented.

### 3.1.2.2 Processing the Virtual World data

When we refer the data coming from the Virtual World, we mean by this, the data virtually received by the synthetic character inhabiting the virtual world, which is
controlled by the user. The information that the Affective Input Component receives from the Virtual World corresponds then to the perceptions of the agent the user is controlling.

This information corresponds to the description of three types of perceptions: events, actions and objects, according with the OCC theory, indicating the agents involved (when involved), the characteristics and the time stamps those perceptions occurred.

The perception information flows directly into the Filter mechanism, which according to the present goals can ignore some of that information.

The concept of the Filter mechanism consists in avoiding the emotional component with information that the attached application does not want to be considered emotionally. Therefore, the filter mechanism, when explicitly indicated by the application, to only filter actions, events or agents with a specific characteristic, creates a rule for that purpose. For this reason, the filter module has also associated a JESS mechanism. The rules produced in the filter module have as their antecedent the description of a generic fact with the referred characteristics. The consequent of these rules are the instantiated facts themselves that correspond to those characteristics. Every fact that enters in the filter module will only be transmitted to the Affective Input Component, if the corresponding filter rule is activated, meaning that the instantiated fact corresponds to the characteristics imposed by the application, to be emotionally considered.

The next process, under which the information already filtered is submitted, is very similar to the ending process of the physical data. The information coming from the virtual world, already translated as facts due the filtering process, is added to the list of facts of another JESS mechanism for the AUM component process, now composed by six jess files corresponding to the six emotions that can be generated by this component. These facts are then used in a cyclic run of this expert system, checking for the rules to be activated, analogously to the physical processing. The activated rules, contrarily to the physical process, produce only descriptions of emotions.

This JESS mechanism also receives the information produced by the physical component describing actions and emotions inferred. Because here is the point where all
the information is gathered and analysed in order to produce emotions, we considered it
the kernel of the affective user model.

3.1.3 Using the AUM Component

The Affective User Modelling Component can handle data coming from both the
Physical Interface and the Virtual World components. In order to the Virtual World
component may interact with this component, the VW must first register itself to the
Affective Input, so that this component can create a dedicated connection to process the
AUM information (physical and virtual information). The registration process is
analogous to the registration of the Physical component. The figure below shows the
sequence diagram for the registration process of the VW component.

![Figure 3.3 - Registration Sequence for the VW component](image-url)
After the registration process, the VW component can then start to send messages containing perception information. These messages must follow the structure defined by the XML schema definition described below.

XML schema definition for the VW Component messages:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="VWMessageXMLSchema">
  <xsd:element name="message">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="msg-id" type="xsd:string" />
        <xsd:element name="time-stamp" type="xsd:string" />
        <xsd:element name="perception" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="type" type="xsd:string" />
              <xsd:element name="attributes" minOccurs="1" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="name" type="xsd:string" />
                    <xsd:element name="action" type="xsd:string" />
                    <xsd:element name="target" type="xsd:string" />
                    <xsd:element name="whom" type="xsd:string" />
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
Below, we show an example of a message produced by the VW component, which follows the above definition. This perception is of type agent. If the perception is of type event or object, some attributes may be omitted.

```xml
<message>
  <message-id>2</message-id>
  <time-stamp>17:20:29</time-stamp>
  <perception>
    <type>agent</type>
    <attributes>
      <name>Agent_A</name>
      <action>give</action>
      <target>key1</target>
      <whom>Agent_B</whom>
    </attributes>
  </perception>
</message>
```

In order to interpret these perceptions, the AUM component is attached to six jess files, corresponding to the six emotions this components can generate: happiness, sadness, anger, disgust, surprise and fear. These files can be edited using the notepad and the rules must once again be written according to JESS syntax as explained above. The structure of the facts describing the events, agents and objects, the types of perceptions, are defined in the initialization.jess file.

This component, according to the rules specified for each class of emotion, will then produce the emotions that were activated in each jess file. This information is then translated to an XML message conveying the emotion characteristics. This message is later sent to the synthetic agent in the virtual world, so that it represents the corresponding emotional state.

### 3.1.4 Defining Events, Agents and Objects in the AUM

According to the OCC theory, emotions are direct or indirect reactions of three aspects of the world: events, agents and objects. Therefore, because we represent the interaction
of the emotional component with a virtual world, we must define a representation of them in the system. These definitions are presented below in JESS syntax.

; The object template
(deftemplate object
  "object structure"
  (slot name)
  (slot feature)
  (slot msg-id)
  (slot time-stamp))

; The agent template
(deftemplate agent
  "agent structure"
  (slot name)
  (slot action)
  (slot target)
  (slot whom)
  (slot msg-id)
  (slot time-stamp))

; The event template
(deftemplate event
  "event structure"
  (slot action)
  (slot what)
  (slot msg-id)
  (slot time-stamp))

These three aspects represent the three different types of perceptions received by the Emotional Component as depicted in the figure below. These perceptions are sent to the affective component, according to the XML definition of a perception described in the previous section, referring the respective aspect in the type field of the message.


3.1.5 **Overall architecture**

In sum, we can describe the AUM component using a class diagram to overview the implemented architecture. The figure below shows the UML class diagram that specifies the classes that compose this component. This picture describes the classes’ properties and their relationships.

![Class Diagram for the AUM Component [UML]](image)

The *EmProcessor* class is the nuclear entity of this diagram. This class is responsible for processing the data coming from both the physical input (the doll) and the virtual world. Data coming from the virtual world assumes the form of perceptions, which can be of...
type Object, Agent or Event. The *Gesture* class, on the other hand, describes the data coming from the physical interface.

Besides the information received from these two input channels, the EmProcessor maintains a list of goals, which is used to infer the current emotion. This entity then generates an *EmotiveAction*, which can be defined, in short, as an action description with an emotion characteristic.
3.2 Affective User Models for James the Butler

The role of James the Butler is to act as mediators between the human and the application (a wine store) and to be capable of personalising the interface by monitoring and sensing user actions, behaviour, product interests, and preferences. James is in effect an Interface Agent (IA), the functionality is realised on two levels: the service level and the interface level. The IA is, hence, considered a service agent that must communicate and negotiate with other agents in a multi-agent system to determine which and how services are to be provided.

Figure 3.6 – James the Butler
3.2.1 Context Sensitive Personalisation

As all software agents are distinguishably characterised by the services they provide, the IA is principally characterised as a user-oriented agent. It is expected to facilitate and provide mechanisms that help enhance an application’s efficiency and usability from both interface and functionality perspectives:

- simplifying and clarifying the user interface, the information presented and the services available,

- providing means for explicit, more efficient user interaction

- providing assistance on understanding the application as whole in terms of:
  - how it functions
  - the services provided
  - personalized focus of services to individual users
  - providing support for user tasks, plans, and intentions
  - presenting information more effectively by personalizing and tailoring the interface
  - interacting and communicating with the user to provide personalized services.

James exhibits aspects of the characteristics described hereafter, so as to assimilate, accumulate and maintain the necessary data within a user profile to provide personalised and context sensitive assistance and recommendation in a clear simplified manner:
Accessible

James is constantly on standby ready to offer assistance when appropriate whether or not it is asked for, and will draw the attention of the user to its existence. At the presentation level James presents itself, its intentions and achievements to the user. The nature or manner of this visualization need not necessarily be manifested in human-like or animated figures the main concern is to draw the user’s attention to the services and achievements of the interface agent. Animated figures and personified manifestations can be considered as an added feature to the interface agent but are not a principle or arguably not an essential characteristic.

The user has control over when to resort to assistance and when not. The user may consult the assistant for help or may delegate routine tasks when required. This approach to agent visibility provides a framework for passing control between the user and the interface agent allowing the user to become more confidant with the interface agent’s abilities and can then build trust in the services, help and information being provided.

Adaptive

The interface agent provides it’s proactive assistant to a specific user by observing and monitoring his/her actions, interests and frequent interactions within a particular application and according to its role. Incrementally, the interface agent learns about user preferences and adapts the interface accordingly. It also uses its observations and learning to become better acquainted and familiar with frequent requests and tasks gradually gaining more competence so as to suggest better or alternative ways to perform these tasks, perform such task autonomously and only update the user of status, results and/or problems.

Autonomous

Autonomy is a key characteristic of James although it is one that the user is usually unaware of; nonetheless, it is one that contributes substantially to the James’s proactive
behavior. It facilitates for providing a higher level of assistance by operating independently and/or concurrently with the user [Wooldridge & Jennings, ‘95], even though this may not always be of value to the user. By constantly operating in the background the interface agent (James) is able to sense and respond to activities in the immediate environment, carry out routine, complex or time consuming tasks on a users behalf only prompting the user when there are results, complications, or when queries or clarifications are required.

**Proactive**

James is integrated with application services with minimum overload on a user’s focus of attention. It needs to be constantly active in the background observing and learning from the environment, user actions, and will naturally act on demand to perform a particular task as instructed, react as a result of problems or irregularities in the system informing the user when appropriate, and be proactive.

**Persistence**

Persistence of context, to provide an extension to personalised information that maintains the user’s current interests. Also, to support off-line information and service gathering and to develop the functionality to continue to work “off-line”, acting proactively or attempting to complete a transaction. This persistence means that each user has an agent identified and working for them, which also continues to work for them whether or not they are actually on-line. Existing and new interface modalities can ensure that the user may make contact with, or be contacted by, their agent by voice, email, or other means.

Persistence is necessary for many of the agent’s most important traits:

- **Reliability**: when a user requests a service the agents need to persist in doing that until it is complete whether successfully or unsuccessfully.
• Personalisation: agents can adapt to customer needs and preferences by updating
  the user profile with a service base and affect state history log.

• Added social traits: agents can remember who the users are and their
  characteristics, thus presenting a consistent persona.

• Learning: agents can modify their behaviour according to experience.

• Mobility: the same agent (in terms of knowledge, behaviour and appearance)
  should appear regardless of location or host computer.

In general, the agents in this demonstrator persist by keeping record of their patterns of
behaviour and interaction, affective state in context with various situation, personality
traits and customer preferences. James maintains its persona, customer awareness and
behaviour following system failures or modifications.

The general benefit of applying the Personal Assistant metaphor lies in the above
characteristics, which collectively may enable the James to enhance an applications
efficiency and effectiveness, reduce the over all work required in using the application,
and provide a more extensible design for the interface developer.

The James the Butler demonstrator is implemented as an agent, which uses an expert
system to manage its internal states and uses the components defined by the SAFIRA
toolkit to for its affective behaviour. The agent maps the semantic of FIPA
performatives to basic JESS (JAVA Expert System Shell) functions. More complicated
commands are communicated to the Database Agent in the content of an FIPA ACL and
executed by the JESS engine. The agent uses profiles to dictate its basic operation style
and the kind of knowledge it works on. For example, on start-up in the system the
James loads a Knowledge base which has domain facts and heuristics about wines.
When a user logs into the system after validation James loads that persons profile, the
agent then uses these two knowledge bases to personalise general questions from the
user to find the appropriate product that best fits the users request and profile. This basic
functionality can also be made to act as a Vendor Agent acting on behalf of the used retailer. It can do this by loading the appropriate profile for a vendor and then it will behave accordingly. Vendor profiles contain information about products held at the vendor store: inventory, prices, discounts and special offers. Its functionality is biased to the vendor and it is the role of the agent is to filter information from available vendors based on user budget specifications and preferences and present the customer with a list of one or more alternatives.

### 3.2.2 User Profile

The user interface represents the sole point of contact between Agent and Customer (User), and a variety of techniques, delineated in this section, are employed to extract valid personalisation information. Some considerable care must be taken if the criteria previously discussed are to be met. Our approach is to combine active, elective and passive information gathering methods.

#### 3.2.2.1 Gathering Customer Preferences

Active methods include electronic form filling. We request as little personal data as possible in this way, as this can be seen as intrusive by some users (decreasing the sense of privacy) and can delay access to the main business oriented activities. Some active information gathering is inevitable, such as name and delivery addresses. In the case of both wine and video sales the customers age band must be ascertained in order to meet legal obligations, fortunately, over 18 years old the actual value is irrelevant. We also request a (self-assessment) of computer experience to tailor the amount of introductory help presented on first use.

In general, we prefer not to present the customer with an extensive “preferences” form, but rather to gather that information incrementally by elective means. Each product selection dialog presents two extra buttons, Like and Dislike. Pressing either of these buttons informs James that the customer prefers (or not) the current selection indicated.
This information then forms part of the persistent knowledge James holds about the customer.

Finally, we may infer preferences from the actions of the customer and the various choices they make. This is at the same time more direct but carries a higher risk than the elective technique. As this profile is built over time, we consider that the investment a customer makes in informing James as to his or her specific likes and dislikes contributes to the sense of loyalty between customer and Agent system. It is clear that these mechanisms must be applied judiciously; the Agent should not appear to act slavishly as a consequence of each preference. It must also allow for exceptions, such as when a customer selects a gift for someone with different tastes, and for the changing tastes of individuals. To address this point the affective state of the customer within context of situation, event, service, time, action, system response and overall system environment is also taken into consideration while maintaining the customer profile. User affective states are inferred through the interactions with James and the consequent services request, as well as through the use of Affectiveware as discussed in section 1.2.2.3 Gathering Customer Affective States.

A main interaction that concerns the James the Butler demonstrator is the business-to-consumer (b2c) interaction involved in retail, in which loyalty is created by human-human interaction and the personal touch. As such, a question this demonstrator addresses is how to reproduce this kind of service online? Everyone remembers personal service or a know-legible shop assistant, and such intelligence and individuality (one-to-one treatment) implies that the multimedia interface could benefit from some kind of similar personality and sensitivity. This is realised in this demonstrator by the implementation of agents with a visual animated life-like character and a distinct personality. With the intention for the customer to be in control: that is the agent is highly intelligent, highly competent character that is essentially a servant. The character adds the personal touch to the interface, but moreover increases in competence over time, thus increasing its intrinsic value to a single customer (i.e. its owner). The
concept is reinforced by an Interface Agent with a visual representation of a butler, ‘James’.

James is an agent that amplifies or modifies the motivational state of an agent and its perceived bodily state. It has the ability to perceive and produce the visual (animated expressions), verbal and non-verbal signals and regulate the flow of information between service agents, the interface agent and the user. These capabilities enable James to engage in complex interactions with customers via natural social communication rather than complex command languages, or direct manipulations.

Thomas and Johnson in the Illusion of Life [Thomas & Johnson, ‘81] discuss how an animated character’s personality consists of characteristic attitudes and actions that users learn to associate with that character, as revealed during their interaction with the system. Based on this, James’s development of it’s characters’ personality is formulated by establishing attitudes and behaviours users will come to expect from the character [Field, ‘94] and which relate to the domain being presented. Hence the reason behind choosing James the butler with a personality that is commonly associated with butlers.

Personality is just one form of creating, likeable characters. Personality and Affective states are represented externally through manifestations and changes in animated character movement and gestures, intonation of conversations whether verbal or textual; and internally in their reactive, proactive and adaptive behaviour.

**Types of Multimodal interactions devices used**

In the James demonstrator implementation we provide services in the following modalities, voice, text, voice and text, animated character with voice. The system facilitates enabling, disabling and combining these modalities to aid the evaluation process. In each case the information presented, the services provided and the recommendations given are all the same (i.e. that applicable to the wine, but derived from the same database source) and the only variations is they channel in which the output is conveyed.
1. Text only output

Here help and recommendations provided by the system are presented to the user in text boxes that are displayed on request. These boxes are in the form of little “Post-it” note boxes that appear in a constant position on the right hand corner of the display. The user can close (remove) the text boxes (Post-its) by right clicking on the box.

2. Voice output

The same help and recommendations provided in the text boxes are prompted to the user through audio channels. The user has to listen to instructions; product descriptions or recommendations to get the information requested but can interrupt the voice message at any time.

3. Voice and text only

Here both the modalities described above are combined. The help and recommendations are presented in text boxes displayed and are read to the user using text-to-speech synthesis. The user can listen to instructions, product descriptions or recommendations and/or can read the information text displayed in the Post-it box at the same time. The advantage here is that the user’s attention is drawn to the box displayed and has the option of either listening to or reading along with the synthesised voice.

4. Character with voice output

Here James is embodied in a synthetic life-like character animation that moves about the screen and can point to objects on the screen providing help, recommendations and product descriptions through audio channels.

We use affect primarily for its influential implications on behaviour, and specifically because it allows the implementation of a social channel of interaction that is fundamental to the effectiveness and believability of a conversation. For this channel to be truly affective, the James character must not only facilitate the channel, but must appropriately respond to cues from the user and the system environment and further, be able to produce affective responses that reinforces the intended message rather than confuse the situation [Bates, ‘94]. An example in this demonstrator would be that the
James will express *happiness* when he has successfully accomplished a task requested by the user, or *excitement* if he has found special offers that match the user’s preferences or requirements.

### 3.2.2.2 Gathering Customer Affective States

Active methods include electronic form filling. We request as little personal data as possible in this way, as this can be seen as intrusive by some users (decreasing the sense of privacy) and can delay access to the main business oriented activities. Some active information gathering is inevitable, such as name and delivery addresses. In the case of both wine and video sales the customers age band must be ascertained in order to meet legal obligations, fortunately, over 18 years old the actual value is irrelevant. We also request a (self-assessment) of computer experience to tailor the amount of introductory help presented on first use.

In general, we prefer not to present the customer with an extensive “preferences” form, but rather to gather that information incrementally by elective means. Each product selection dialog presents two extra buttons, Like and Dislike. Pressing either of these buttons informs James that the customer prefers (or not) the current selection indicated. This information then forms part of the persistent knowledge James holds about the customer.

Finally, we may infer preferences from the actions of the customer and the various choices they make. This is at the same time more direct but carries a higher risk than the elective technique. As this profile is built over time, we consider that the investment a customer makes in informing James as to his or her specific likes and dislikes contributes to the sense of loyalty between customer and Agent system. It is clear that these mechanisms must be applied judiciously; the Agent should not appear to act slavishly as a consequence of each preference. It must also allow for exceptions, such as when a customer selects a gift for someone with different tastes, and for the changing tastes of individuals. To address this point the affective state of the customer within context of situation, event, service, time, action, system response and overall system environment is also taken into consideration while maintaining the customer profile.
User affective states are inferred through the interactions with James and the consequent services request, as well as through the use of Affectiveware as discussed in section 1.2.2.3 Gathering Customer Affective States.

A main interaction that concerns the James the Butler demonstrator is the business-to-consumer (b2c) interaction involved in retail, in which loyalty is created by human-human interaction and the personal touch. As such, a question this demonstrator addresses is how to reproduce this kind of service online? Everyone remembers personal service or a know-legible shop assistant, and such intelligence and individuality (one-to-one treatment) implies that the multimedia interface could benefit from some kind of similar personality and sensitivity. This is realised in this demonstrator by the implementation of agents with a visual animated life-like character and a distinct personality. With the intention for the customer to be in control: that is the agent is highly intelligent, highly competent character that is essentially a servant. The character adds the personal touch to the interface, but moreover increases in competence over time, thus increasing its intrinsic value to a single customer (i.e. its owner). The concept is reinforced by an Interface Agent with a visual representation of a butler, ‘James’.

James is an agent that amplifies or modifies the motivational state of an agent and its perceived bodily state. It has the ability to perceive and produce the visual (animated expressions), verbal and non-verbal signals and regulate the flow of information between service agents, the interface agent and the user. These capabilities enable James to engage in complex interactions with customers via natural social communication rather than complex command languages, or direct manipulations.

Thomas and Johnson in the Illusion of Life [Thomas & Johnson, ‘81] discuss how an animated character’s personality consists of characteristic attitudes and actions that users learn to associate with that character, as revealed during their interaction with the system. Based on this, James’s development of it’s characters’ personality is formulated by establishing attitudes and behaviours users will come to expect from the character.
[Field, ‘94] and which relate to the domain being presented. Hence the reason behind choosing James the butler with a personality that is commonly associated with butlers.

Personality is just one form of creating, likeable characters. Personality and Affective states are represented externally through manifestations and changes in animated character movement and gestures, intonation of conversations whether verbal or textual; and internally in their reactive, proactive and adaptive behaviour.

3.2.2.3 Maintaining Customer Profiles

In the James the Butler demonstrator profiles are used by agents to take on certain aspects of the role of retailers and customers. The idea is that a customer has a buying history which can be used to form the basis of a profile of that customer’s preferences for certain products and services. The customer is encouraged to view and change this profile so that it can better reflect the customer’s preference. An agent uses this profile along with domain knowledge about the products available in the system. This means a consumer can ask very general questions about the products and the agent will filter them according to the consumer’s profile and its domain knowledge. It can then search the product databases for specific products that match the filtered request.

This can be made more realistic if other agents in the system have profiles that enable them to work on behalf of the vendor. The vendor can have information of a particular customer’s purchasing preferences from looking at previous purchases the customer has made. The vendor agents can then make suggestions to the consumer about offers and deals tailored to its perception of what s/he wants. Naturally, as the customer buys more the vendor agent will have more data to reason about the perceived preferences and so can tailor its offers better to meet customer needs.

Given this scenario it is not hard to see that agents can then negotiate on a customer’s behalf. A customer indicates that they are interested in a type of product. A search of the system according to the personal profile returns to the customer a certain number of choices. However a vendor agent might be able to make a number of offers or deals to
the customer trying to get the customer to buy products it wants to sell but are at the same time relevant to the customer. At this point the vendor and agent and the customer agent could negotiate using the customer’s preference on how much he or she are prepared to spend and the vendors preferences for increasing gross margin and selling products that need pushing. A compromise is reached and the customer is given a number of choices.
4. Conclusions

This deliverable focuses on extending current user modelling techniques to represent affective states of the user. In here, we specify and describe issues associated with the development of a shell for affective user modelling.

Then, this deliverable provides the description of the Affective User Modelling component that will be integrated in the Toolkit and used in two of the Safira applications: FantasyA and James the Butler.

Note that this deliverable should be seen as a draft version of the final document because there is work on WP3 still under development, in particular all aspects related with the integration and use in the demonstrators.
Bibliography


