KEER2010, PARIS | MARCH 2-4 2010
INTERNATIONAL CONFERENCE ON KANSEI ENGINEERING AND EMOTION RESEARCH 2010

INTELLIGENT AGENTS FOR PERSONALISED VEHICLE CONFIGURATION

Pavlos SPANIDIS[a], Evangelos BEKIARIS[*a] and Maria GEMOU[a]

[a]Hellenic Institute of Transport, Centre for Research and Technology Hellas, Greece

ABSTRACT

This paper introduces the agent-based framework that exploits the results coming from the Data Mining techniques implemented in the context of the CATER EU FP6 funded project (035030) [1], in order to facilitate mass customization of vehicles in the automotive industry.

CATER aimed to deploy intelligent agents and knowledge management based N-Business systems, that will support modular personalisation of the automotive products to the explicit and emotional needs and wants of the international clients, as well as mass customisation of product fulfillment and flexibility of services to meet the needs of customers, suppliers, sales, marketing and the concurrent engineering team.

Mass production practices manage to reduce manufacturing costs, however they fail to sufficiently reflect personalised customer needs and demands to the final products. The CATER agent-based framework aims to support the provision of more personalised products to automotive industry customers, while preserving the advantages of mass production. Today, only few automotive industries have deployed mass customization systems in their product design and manufacturing processes. None of them combines agents and Data Mining (DM) techniques for mass customisation of vehicles.

The system presented in this paper, proposes a mass customisation system, which applies a set of DM techniques on data from automotive industry customer surveys aiming to generate a set of business rules that associate the users' preferences and affective needs to design elements of vehicles, following the citarasa principles, that are finally extracted into a business-logic component of an agent, representing the automotive industry and seeking iteratively customers' feedback [2].

Keywords: *agents, vehicle personalisation, mass customisation*

[*] Correspondent author: 6th km Charilaou – Thermi Rd., P.O.Box: 60361, P.C.: 57001, Thermi, Thessaloniki, Greece, abek@certh.gr.

1. INTRODUCTION

The main objective of this document is to describe the work done for designing and developing the architecture of the agent – based system developed within the framework of CATER project. The agent – based system of CATER interacts with different modules of the CATER system in order to support the mass customization supply chain including suppliers, factories, subcontractors, warehouses, distribution centres and retailers.

These are namely the following: the Data Mining (DM) module which is part of the Citarasa Engine and the Do It Yourself Design (DIYD) Engine. Citarasa Engine is an Engine that contains a new research focus and perspective that integrates cognition/thinking and emotion/affect in uncovering customer needs [2]. The DM module demonstrates a classifier of customer needs to design parameters of vehicles. The agents interact with the DM module and support its operations with the aim of performing accurate predictions. The DIYD Engine is the application responsible for the interaction of the customer – product planner with the system. The DIYD is the medium that controls and rules the communication of the external users with the Citarasa Engine, the DIYD database, etc. The DIYD database is a database that includes a customization structure and functionality with exemplary product data for vehicle configuration and a web user interface for mass customization, powered by citarasa technology, to support the customer in vehicle configuration tasks, taking into account emotional and functional criteria [2]. All the above modules are interconnected by an application which is called Agent.

In programming, agents are called all the threads that constitute a system [3]. Moreover, each thread can have more than one activities (which are called behaviours) and can execute them concurrently. The developed agent system is responsible for addressing all the requests that it collects during a communication. In more detail, the agent identifies the receiver from its list according to the content of the message and it redirects the message. Additionally, the agent is responsible to return the answer of the receiver to the application that made the request. The agent's clients can be other modules of the main system, CATER system in this case, or third party applications.

Section 2 of the current paper presents the framework of an agent – based system. It provides the theory which is necessary for the better understanding of the agents as a mechanism and the agents – based system of CATER as an application.

Section 3 of the current paper provides the in depth details of the designed agent of CATER. It presents all the functionality of the agent together with the detailed background of the application that has been used for the accomplishment of the work. The mechanism implemented for the accurate prediction of the most suitable vehicle configuration according to the profile and needs of the registered customer is being discussed.

Section 4 concludes the work presented.

2. AGENTS FRAMEWORK

As aforementioned, the main objective of the agent – based system in CATER is to support the connectivity and communication of CATER databases with some other modules of CATER (i.e. the Do It Yourself Design – DIYD module).

In order to support the above functionality, the agent – based system has been developed with a new technology of JADE which is called Web Service Integration Gateway (WSIG). The objective of WSIG is to expose services provided by agents and published in the JADE framework as web services, though giving developers enough flexibility to meet specific requirements. The process involves the generation of a suitable WSDL for each service-description registered with the Data Framework and also the publication of the exposed services in a UDDI registry. The Web Services are becoming one of the most important topics of software development and a standard for interconnection of different applications.

The WSIG add-on of JADE supports the standard Web Services stack, consisting of WSDL for service descriptions, SOAP message transport and a UDDI repository for publishing Web Services using Models [4]. As shown in WSIG Architecture

, WSIG is a web application composed of two main elements:

- the WSIG Servlet
- the WSIG Agent

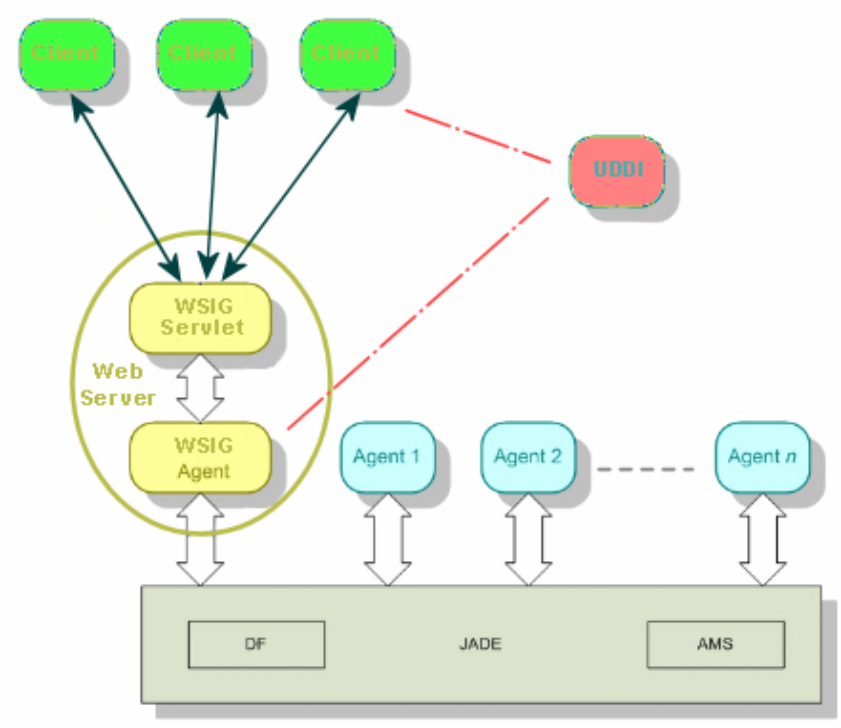

Figure 1: WSIG Architecture

The WSIG Servlet is the front-end towards the internet world [4] and is responsible for:

- Serving incoming HTTP/SOAP requests;
- Extracting the SOAP message;
- Preparing the corresponding agent action and passing it to the WSIG Agent Moreover once the action has been served;
- Converting the action result into a SOAP message;
- Preparing the HTTP/SOAP response to be sent back to the client.

The WSIG Agent is the gateway between the Web and the Agent worlds [4] and is responsible for:

- Forwarding agent actions received from the WSIG Servlet to the agents actually able to serve them and getting back responses.
- Subscribing to the JADE DF to receive notifications about agent registrations / de-registrations.
- Creating the WSDL corresponding to each agent service registered with the DF and publishes the service in a UDDI registry if needed.

Two main processes are continuously active in the WSIG web application:

1. The process responsible for intercepting DF registrations/de-registrations and converting them into suitable WSDLs. As mentioned, this process is completely carried out by the WSIG Agent.
2. The process responsible for serving incoming web service requests and triggering the corresponding agent actions. This process is carried out jointly by the WSIG Servlet (performing the necessary translations) and the WSIG Agent (forwarding requests to agents able to serve them).

The FIPA (Foundation for Intelligent Physical Agents) compliant JADE/LEAP platform [3] adopted allows for an architecture that is:

- Distributed (different platforms);
- Standards based (FIPA, HTTP, XML, RDF);
- Process centric (agents);
- Widely used in ICT (Information and Communication Technologies);
- Open source (possibility of features addition);
- Cross-platform (Operating System, e.g. Linux);
- Variety of message transport protocols.

According to the developers of JADE [3], JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It aims at the development of multi-agent systems and applications confirming to FIPA standards for intelligent agents. It includes:

- A runtime environment where JADE agents can "live" and that must be active on a given host before one or more agents can be executed on the host.
- A library of classes that programmers can use to develop their agents.
- A suite of graphical tools that allows administrating and monitoring the activity of running agents.

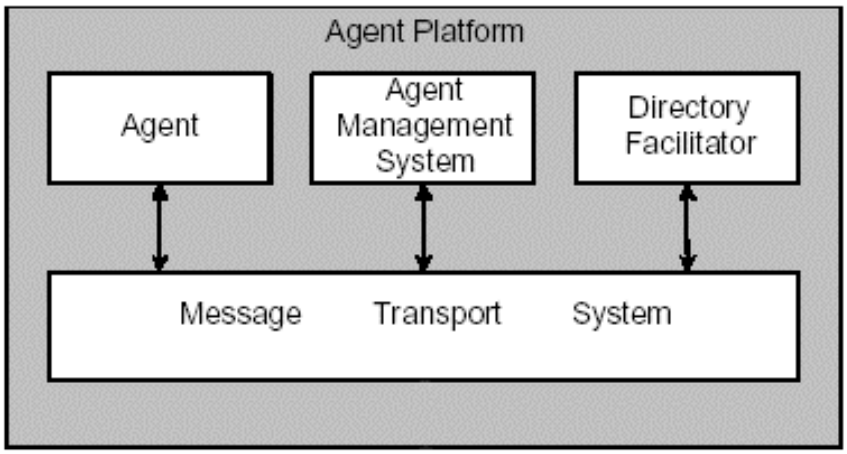

Figure 2: Agent Platform internal diagram

Each running instance of the JADE runtime environment is called a 'Container' as it can contain several agents. A set of active containers is called a 'Platform'. A single special Main Container should always be active in a platform and all other containers register with it when they start.

The Main Container differs from normal containers in the ability of accepting registrations from other containers. This registration can be done by the two special agents that start when the main container is launched. These are:

- The Agent Management System (AMS) that provides the naming service and represents the authority in the platform. The Agent Communication Channel (ACC) is the agent that provides the path for basic contact between agents inside and outside the platform.
- Standards The Directory Facilitator (DF) that provides a Yellow Pages service by means of which an agent can find other agents providing the required services. The standard specifies also the Agent Communication Language (ACL). Agent communication is based on message passing, where agents communicate by formulating and transmitting individual messages to each other.

3. CATER AGENT-BASED SYSTEM

3.1. Agents in the overall CATER Architecture

As it has already been mentioned, the CATER architecture is based on agents. Figure 3 shows the connectivity of the agents with the rest modules of the system.

More analytically, the agent is interconnected with three main modules. These are namely: a) the Web interface, b) the citarasa engine and c) the DIYD engine of the system. On the Web interface, the agent allows the user to register and/or login him/herself to the system. Specific entries are requested by the user, such as the name and surname of the user, desired username and password and also the occupation, the region, the age and the gender.

The occupation, the region, the age and the gender in specific, constitute the input that is required by the citarasa engine in order to predict a vehicle configuration, customized to the specific user, classified also per citarasa descriptor (i.e. "comfortable", "professional", "sporty", etc.). The prediction of the most suitable vehicle configuration is performed through data mining techniques that utilise the respective user information mapped to user preferences that have been collected in the past and have been deployed for the training of the system (Data Mining module), whereas all requested entries by each registered user are collected in the citarasa engineering database.

The citarasa engineering database currently consists of 618 entries/semantic descriptors [coming from customers surveys from 10 countries - Asia (Malaysia, Singapore), North Europe (Finland, Sweden, Switzerland), Central Europe (France, Germany, UK), and South Europe (Italy, Greece), addressing both car and truck segments, and all age groups (18-24 years), (25-54 years), (55-65 years)].

Finally, the agent is interconnected with the DIYD engine that constitutes the interface of the user with the citarasa engine. The DIYD engine requests from the user (via the web interface) to choose among a list of citarasa descriptors (i.e. "comfortable", "professional", "sporty", etc.) that according to his/her opinion characterize in the most felicitous way his/her overall preference regarding the vehicle s/he wishes to view and further configure. The DIYD engine, utilising the output of the citarasa engine, finally provides to the user a suggested vehicle configuration, customized to his/her profile and declared preference.

It should be noted that the agent has been designed in such a way so as to support also the self training of the system. Every time a user completes his/her vehicle configuration process, the CATER agent stores this information. A specific number of new entries on the database trigger the update of the data mining rules that are responsible for the vehicle configuration prediction recommended to the user.

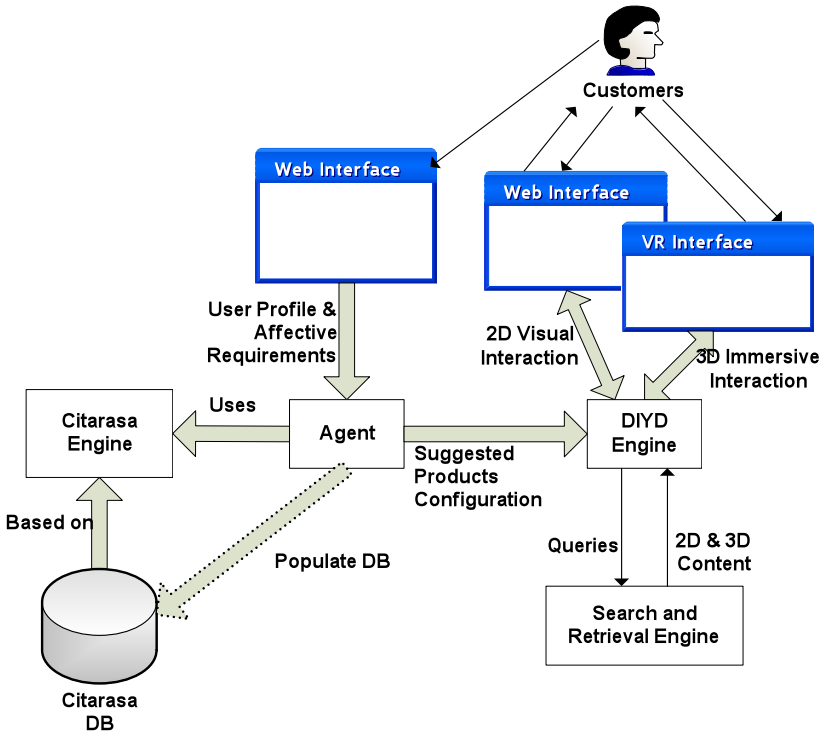

Figure 3: Agent's conceptual architecture

3.2. CATER Agent-Web service description

The agent – based system consists of a number of functions. Each function is responsible for a particular activity and these activities are accessible through a special XML file, the WSDL file. In this file, the client is able to find all the available activities that the agent can perform. **Error! Reference source not found.** below contains the list of the actions that the CATER agent performs.

Table 1: List of functions of the agent – based system of CATER

Activity	Function	Description
Registration	setUser( )	This function has a list of attributes as input (name, surname, username, password, region, occupation, age, gender) and outputs "0" or "1" (false or true) which indicates the successful addition of the data in the database.
Registration	getUsernameExistance( )	This function examines if a specific username already exists in the database. It has one attribute as an input, the username and it outputs "1" or "0" for true or false respectively.
Log In	getUserId( )	This function provides the ID number of the user. It has two attributes as input (username and password) and outputs the ID number of the user when the log in is successful or "1" if there is an error with

Activity	Function	Description
		the username and/or password.
Log In	updateUser( )	This function is responsible for the user profile update. It gets one value as input (the username) and it returns "0" or "1" (false or true) which indicates the successful addition of the data in the database.
Accessing CATER list	setComponentValue( )	This is a function that stores the user's history. It keeps the CATER components database updated depending on the choices-selections of the user. The function stores the user's updates on a specific component with input attributes user ID, region, occupation, descriptor, component, component ID, attribute and outputs "1" or "0" (false or true) indicating the successful addition of the data in the table.
Accessing CATER list	getUserAttributes( )	This function collects all the attributes of a user. It requires the ID number of the user as input and it outputs a vector (an array of data) with the name, surname, username, password, region, occupation, age, gender.
Accessing CATER list	getComponentId( )	A function that returns the ID number of the vehicle-component combination according to the vehicle type (i.e. car or truck) and the component name (i.e. mirror, steering wheel, etc.).
Accessing CATER list	getActualAttribute( )	This function uses semantics (guided by ontology) for the mapping of predicted design parameters to actual elements which are available in the DIYD engine. It has two input attributes (the component ID and the extracted output) and returns the corresponding element of the DIYD Engine.
Accessing CATER list	getStatistics( )	This is a function that provides the percentage of the available choices of a descriptor according to the entered input (region, gender, age range and descriptor).

These functions are available through the World Wide Web. Any module that needs to interact with the CATER system has to follow the rules of the above functions in order to retrieve the required/requested results.

4. CONCLUSIONS

The current paper describes the design and the development of a specific module of the CATER integrated system, namely the agent – based system of CATER which is responsible for the interconnection and interface of different modules of the system, aiming, finally, at proposing a personalised vehicle configuration to the user. It should be noted that the user is able to further elaborate the proposed by the system vehicle configuration through the CATER configurators, if wishes so.

The agents' technology deployed is a FIPA compliant JADE/LEAP platform technology which has been indicated as the best solution for Client – Server communications [3]. The functionality of the agent required the use of another add-on application of JADE which is called WSIG [4]. This add-on transformed the agent's functionality to web service in order to be available to anyone through the World Wide Web and made feasible the interface of CATER engines output through a web interface.

The personalisation enabled by the CATER agent lies in the consideration of specific information (age, region, gender, occupation) of the registered user as well as declared preferences (reflected through the citarasa descriptors), which are then communicated to the respective back-end modules of the system, that in turn predict and compose the most suitable vehicle configuration for each user. The communication of this result to the user is again a responsibility of the CATER agent system.

A valuable advantage of the CATER agent system is the ability to store the history of each user vehicle configuration tried. The history records are then utilised for the update of the prediction rules, and as such, of the progressing improvement of their accuracy.

ACKNOWLEDGEMENTS

Work reported in this paper was partially funded by the EU funded project CATER (http://www.cater-ist.org/), contract number: IST-035030.

REFERENCES

1. Annex I-"Description of Work", CATER project, CN. 035030, Sixth Framework Programme, Priority 2-IST, Information Society Technologies.
2. Khalid, H.M., and Helander, M.G., Customer emotional needs in product design, *International Journal on Concurrent Engineering: Research and Applications*, 14(3), 197-206, 2006.
3. Jade Agent DEvelopment Framework – an Open Source platform for peer-to-peer agent based applications, <*http://jade.tilab.com*>, 2008 [Accessed 2008 September 6].
4. Jade Web Services Integration Gateway (WSIG) Guide, PDF file, <*http://jade.tilab.com*> (add-ons area), 2008 [Accessed 2008 September 10].