A DIYD (Do It Yourself Design) e-Commerce System for Vehicle Design Based on Ontologies and 3D Visualization

L. Makris, N. Karatzoulis, and D. Tzovaras

Informatics & Telematics Institute
1st Km Thermi-Panorama Road, PO Box 361,
GR-57001 Thermi-Thessaloniki, Greece
{lmak, nkaratz, tzovaras}@iti.gr

Abstract. The state of the art in vehicle configuration is still very much characterized by a face-to-face sales situation. In addition, web browsers are becoming market places. But direct sales over the internet, without contact with a sales person constitute still a small segment of the market, of only a few percent for European manufacturers. The internet is more used as a medium to gather information. A standardised DIYD vehicle configuration is thus a must for European manufacturers today. This paper presents an Intelligent DIY e-commerce system for vehicle design, based on Ontologies and 3D Visualization, that aims at enabling a suitable representation of products with the most realistic possible visualisation outcome. The platform, designed for the automotive sector, includes all the practicable electronic commerce variants and its on-line product configuration process is controlled by an ontology, that was created using the OWL Web Ontology Language.

1 Introduction

This paper presents an intelligent and user-friendly e-commerce solution, by adopting additional technologies, such as a configuration utility, supported by an intelligent help desk system and 3D visualisation in a virtual reality environment. The focus of this paper is on the automotive sector; however the intention is that the system to be developed will be suitable for suppliers, and wholesalers, from other sectors, such as clothing, bicycles, etc. The conception of the system is as generic as possible, in order to facilitate knowledge transfer to other industrial sectors.

The technical contribution of this paper is to present how to: (i) integrate and develop a product Visualisation Tool, capable of handling different media types (2D and 3D), (ii) introduce a novel configuration module, which is capable of integrating different functionalities, input and output devices, and (iii) provide enhanced workflow mechanisms and tools for an easy integration into modern legacy systems (ERPs).

The implementation of this work has started in the context of the CATER European project (CN 035030), funded under the 6th EU framework, which aims to aims to deploy intelligent agents and knowledge management based N-business systems, that will support modular personalisation of the automotive products to the explicit as well as emotional needs and wants of the international clients, as well as mass customisation and reverse engineering of product fulfilment and flexibility of services, to meet the needs of customers, suppliers, sales, marketing and the concurrent engineering team and, within this framework, to develop a customisation database structure and functionality with exemplary product data for vehicle configuration.

The described platform, called hereafter CATER platform, has been designed for industrial automotive design products, that can be configured on-line. The main modules of the CATER platform are the **Intelligent Configurator Module** (and its ontology structure) and the **3D Visualisation Module**. These two modules combined, comprise an advanced automotive design system, that offers different products and services to customers, based on their demand and not on a fixed product line.

The paper is structured as following: in section 2 the intelligent configurator module is presented, with special focus on its architecture and ontology structure. In section 3 the 3D visualization module is analysed and in section 4 the e-shop solution and its integration with existing ERPs, is discussed. Finally, section 5 presents some experimental results and concludes the paper.

2 Intelligent Configurator Module and Ontology

The Intelligent Configurator module is a web based application, that allows the user to assembly vehicles, based on the available vehicle parts, that are being stored in the systems repository, maintained by the vehicle manufacturer. Figure 1 displays the Units of the Configurator module.

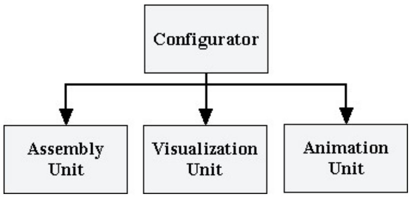
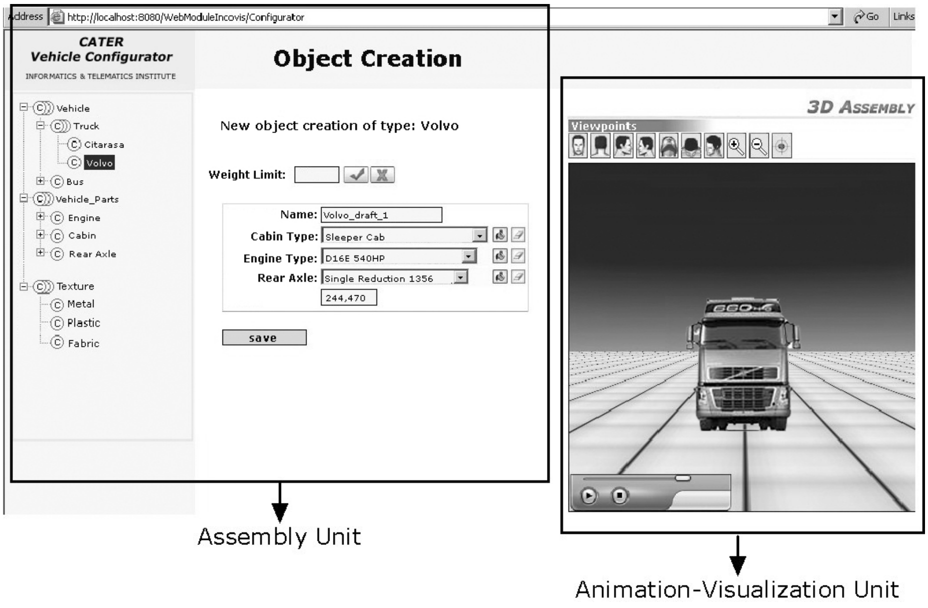


Fig. 1. The Units of the Configurator Module

The Assembly Unit allows the user to insert individual 3D objects to the scene, that can consist of a fully functional vehicle. The user can compose the desired vehicle according to his/her needs, by selecting the vehicle’s parts. The vehicle part and the texture selection processes are being controlled by the restriction mechanisms, that are generated from the system Ontology [1, 2, 3]. The main functionalities of the Assembly Unit are the following: (i) Insertion of 3D object parts, (ii) Selection of texture and (iii) Assembly process based on rules (e.g. weight).

The purpose of the Visualization Unit is to record and store the 3D object assembly steps in real-time. The assembly sequence is being stored in the 3D animation repository for future reproduction.

The Animation Unit allows the reproduction of the vehicle parts assembly processes, that are stored in the animation system repository. In the Animation Unit the user can control the viewpoints and the playback of the loaded vehicle assembly process. The web interface of the Configurator Module is depicted in Figure 2.



New object creation of type: Volvo

Weight Limit:

Name: Volvo_draft_1
Cabin Type: Sleeper Cab
Engine Type: D16E 540HP
Rear Axle: Single Reduction 1356
244,470

Fig. 2. The web interface of the Configurator

2.1 Configurator Architecture

The Configurator is implemented (Figure 3) using Java programming language. The system runs on Apache Jakarta Tomcat [4], as Java Servlet, and is based on the Jena framework [5], which is a Java framework for building Semantic Web applications.

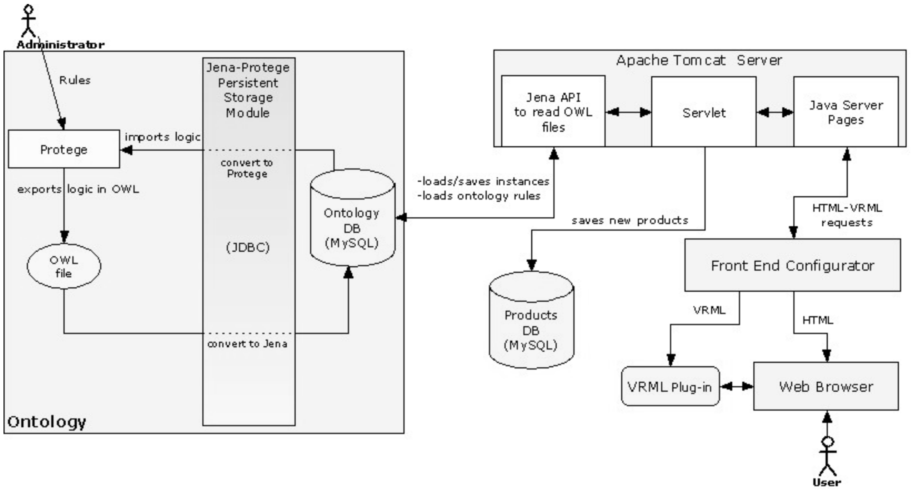


Fig. 3. Configurator Architecture

The ontology is created using Protégé [6], which is an open source knowledge-base framework. The persistent store of the ontology is achieved using the persistence subsystem of Jena, while the 3D visualization was developed using the VRML [7] standard and External Authoring Interface (EAI) mechanisms.

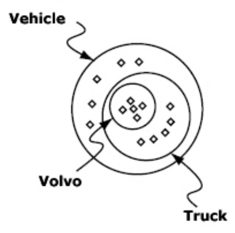
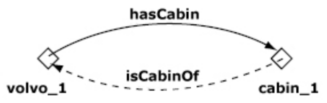
2.2 Ontology Development

The ontology was created using the OWL Web Ontology Language [8], and the Protégé OWL-Plugin [9], which is an extension of Protégé, with support for the Web Ontology Language (OWL).

The OWL-DL profile, used in order to create the ontology, is based on Description Logics. Description Logics are a decidable fragment of First Order Logic2 and are therefore amenable to automated reasoning. It is therefore possible to automatically compute the classification hierarchy and check for inconsistencies in an ontology that conforms to OWL-DL [10].

The classes of this Ontology are interpreted as sets that contain individuals. They are described using formal descriptions, that state precisely the requirements for membership of the class. For example, the class "Vehicle" contains all the individuals that are of type Vehicle in the CATER domain. The taxonomy of the classes is being achieved using the superclass-subclass model hierarchy.

Table 1. Example of the class hierarchy of the class "Vehicle" and an example of an Object Type Property for the individual "vehicle_1"

Class Model	Object Type Property
	

There are two types of properties supported by our ontology a) Data Type Properties and b) Object Type Properties. These OWL Properties represent relationships between two individuals.

In OWL, properties are used to create restrictions. In our ontology, restrictions were used to restrict the individuals that belong to a class. We used the universal quantifier ∀ restrictions to constrain the relationships along a given property to individuals that are members of a specific class. For example, the universal restriction ∀ hasCabin cabin_1 describes the individuals, whose hasCabin relationships are members of the class Cabin.

Cardinality restrictions were used to define the order in which the individual object parts should appear during the 3D assembly process (i.e. real-time animation). The cardinality restrictions provided the way to describe the class of individuals that have at least, at most or exactly a specified number of relationships with other individuals or datatype values.

The hasValue restrictions, denoted by the symbol ∍, were used to describe the set of individuals that have at least one relationship along a specified property to a specific individual. For example, when we wanted to predefine the dimensions of an individual object, we used a hasValue restriction (dimensions ∍ “40-50-80”).

2.3 Ontology Reasoning

Ontology reasoning was achieved using the Jena OWL reasoner (Figure 4). The Jena OWL reasoner can be described as instance-based reasoner, that works by using rules to propogate the if- and only-if- implications of the OWL constructs on instance data. Reasoning about classes is done indirectly - for each declared class a prototypical instance is created and elaborated. The sub-class and sub-property lattices are cached using the embedded OWL reasoner. Each domain, range, sub-property and sub-class declaration is eagerly translated into a single query rewrite rule. The result of a query to the graph will be the union of the results from applying the query, plus all the rewritten versions of the query to the underlying graph [11].

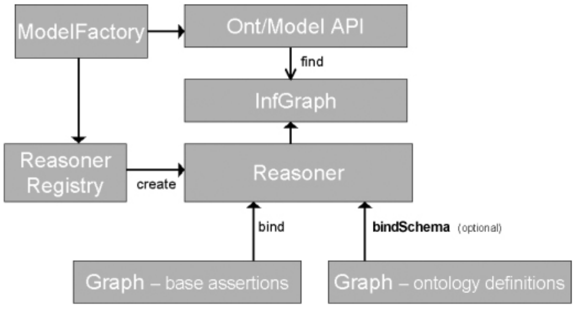


Fig. 4. The Jena Inference API layering [12]

3 3D Visualisation Module

The 3D Visualization Module is realized on the Visualization and Animation Units. The structure of the individual 3D scenes, supported by the 3D Visualization Module, can be seen in Figure 5.

The 3D scene contains the viewpoints, the lighting of the 3D world, the background, the 3D objects and the object functionalities (interactions), that are created dynamically, according to the ontology specifications. The user interaction with the 3D scene is achieved by the use of predefined VRML Protos. For every 3D object that is inserted in the 3D scene an animation, representing its assembly process, is dynamically generated.

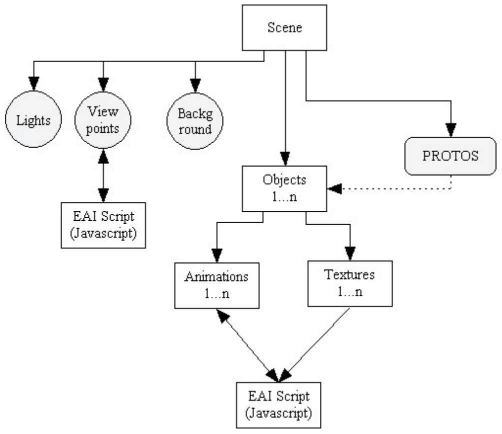


Fig. 5. The 3D scene structure of the 3D Visualization Module

The playback functionalities of the assembly process are controlled by a panel (Figure 6), that was developed using several VRML sensors (TouchSensor & PlaneSensor).

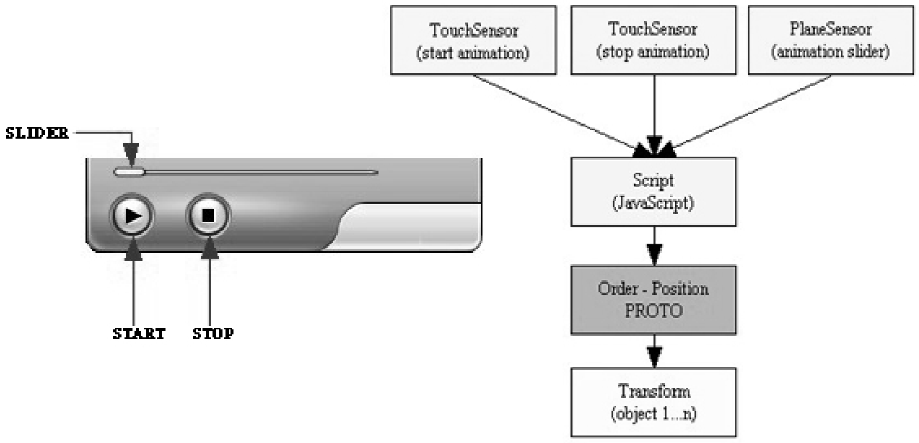

TouchSensor (start animation)
TouchSensor (stop animation)
PlaneSensor (animation slider)
SLIDER
Script (JavaScript)
Order - Position PROTO
Transform (object 1...n)
START
STOP

Fig. 6. The assembly process control

4 The e-Shopping Platform in Practice

The use of the CATER platform "brings" advantages to both suppliers and buyers, regarding (i) the cutback of transaction costs, (ii) the use of automated supply procedures, (iii) economies of scale, (iv) wide access on both local and international markets, (v) dynamic real-time price mechanisms/modules and (vi) the use of compatible/expandable technologies.

The requirements of the described CATER platform for automotive products, that together with the Intelligent Configurator Module and the 3D Visualisation Module comprise the advanced 3D Shop system, are:

- search and present all the available products, based on multi-criteria search engines;
- group products into multilevel categories (set by the e-shop administrator);
- make offers/ sales and promote them;
- update both the product catalogue and all items' availability (set by the e-shop administrator);
- create/use shop baskets (by the end buyers);
- provide several convenient pay/ receive methods;
- provide a secure e-payment credit card transaction (with the use of HTTPS and SSL protocols).

However, the efficiency and overall quality of an e-commerce service depends "heavily" on its automatic connection with the existing ERP (Enterprise Resource Planning) system for the catalogue, prices, stock and product update. In order to integrate all the available ERP data with the e-shop database, a powerful staging mechanism is developed and securely transfers all necessary data. This staging process uses a smart "track changes" algorithm, to enhance the update speed.

There are two staging processes, Real Time Staging and Off Line Staging (that uses an automated batch process). The characteristics of the two staging "methods" are compared in the following table.

Table 2. Staging Procedures Comparison

	Real Time Staging	**Off Line Staging**
Data Update	(+) All data are updated at all times	(-) All data are updated at specifically defined time periods
Infrastructure	(-) Reliable, high speed, technical infrastructure is necessary, available on a 24x7x365 basis	(+) Not so advanced technical infrastructure is necessary
Security	(-) The system can be secure but certain "protective" actions must be taken	(+) Security is obvious

The previous table shows that a real time staging procedure should be followed only if the nature of the commodity traded imposes the constant database update. In our case, an every day off line procedure is chosen for both security and convenience reasons.

Yet, if we try to deduct a general case example we must notice that each company's and product's needs, concerning the use of an e-market, are different; therefore the connectivity solutions (between an e-shop and an ERP) provided vary, depending on: (i) the ERP used (it can be a widely used international ERP such as SAP, Oracle Applications, etc. or it can be a custom made system, that fits to specific needs), (ii) the transaction volume and the form of the data transferred, (iii) the importance of the information transferred (regarding time, safety etc. aspects), (iv) the use of unilateral or bilateral communication and (v) whether it is an on-line or a batch transfer of data.

5 Experimental Results and Conclusions

E-commerce services offered through a B2C (business to consumer) or B2B (business to business) system, provide the necessary infrastructure for real time e-business and an added value package of services, that guarantee faster and more efficient buy and sell transactions, access to a broadened database of buyers/suppliers and business opportunities, through the development of new partnerships.

In conclusion, in this paper we presented an interactive and user-friendly e-commerce solution for the vehicle sector, but appropriate for other sectors as well. Volvo Technology Corporation (VTEC) has been the end-user responsible for using and testing the CATER platform, so a number of its vehicles were integrated in the platform for evaluation and testing purposes.

Finally, the main contribution is that our approach integrates additional technologies, such as an intelligent configurator module and a 3D visualisation environment aiming at enabling a suitable representation of products, in order to achieve the most realistic possible visualization and simulate a realistic shopping procedure.

Acknowledgements

We wish to acknowledge to the CATER project Consortium for their valuable contributions to this work. The CATER project is partially funded by the EC.

References

1. Kompatsiaris, I., Mezaris, V., Strintzis, M.G.: Multimedia content indexing and retrieval using an object ontology. In: Stamou, G., Kollias, S. (eds.) Multimedia Content and the Semantic Web: Methods, Standards and Tools, pp. 339–371. Wiley, Chichester (2005)
2. Tsampoulatidis, I., Nikolakis, G., Tzovaras, D., Strintzis, M.G.: Ontology Based Interactive Graphic Environment for Product Presentation. In: Proc. CGI 2004, pp. 644–647, Heraklion, Crete, Greece (June 2004)
3. Mezaris, V., Kompatsiaris, I., Strintzis, M.G.: An Ontology Approach to Object-Based Image Retrieval. In: Proc. IEEE International Conference on Image Processing (ICIP 2003), Barcelona, Spain, vol. II, pp. 511–514 (September 2003) http://www.iti.gr/db.php/el/publications/details/532.html
4. Apache Jakarta, http://jakarta.apache.org/
5. Jena, A Semantic Web Framework for Java, http://jena.sourceforge.net/
6. Protégé, An Ontology Editor and Knowledge-base Framework, http://protege.stanford.edu

7. Web 3D Consortium, VRML Standard, http://www.web3d.org/x3d/vrml/
8. Protégé OWL Plugin, http://protege.stanford.edu/plugins/owl/
9. The OWL Web Ontology Language, http://www.w3.org/TR/owl-features/
10. Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C., Practical, A.: Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, Edition 1.0, The University of Manchester (August 2004)
11. Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations, Digital Media Systems Laboratory, HP Laboratories Bristol (2003)
12. Jena 2 Inference Support, http://jena.sourceforge.net/inference/