INTEGRATING ONTOLOGIES AND USER INTERFACES
The XGC Approach

Vicente R. Tomás López[1], J. Javier Samper[2,] Juan José Martínez.[2]
[1] *University Jaume I, Spain,*
[2] *Robotics Institute. University of Valencia, Spain*

Keywords: User Interfaces, Graphics, Intelligent transport Systems.

Abstract: In domains where several and different systems are working jointly, the integration of them via graphic interfaces is necessary to improve their employ and the user acceptance degree. In the transportation domain the interfaces are basically based on object control that allows to the road operator to manage the different traffic elements. The most usual interfaces are proprietary and they use different vocabularies to work with the same concepts. In this paper traffic ontology is presented. It is used to develop a Generic interface: the eXtensible Graphic Container-XGC. This user interface allows to create a standard interface able to integrate different systems. The system presents several good features: 1) traffic ontology is used to develop the object systems and to provide semantic knowledge; 2) the container is developed in Java which allows to use it in different platforms; 3) open source approach.

1 INTRODUCTION

In domains where object control is needed, a graphic user interface is the most common type of interface used at the moment. The modern window-based systems with standardized screens, buttons, etc. provide a small learning curve and a high user acceptance. Unfortunately the development of a user interface is not standardized and between operating systems (MS-Windows, Linux etc.) there are big differences in appearance. So in a heterogeneous computing environment, an operator still needs to learn different interfaces.

Especially in the transportation domain there are big differences in supplier provided equipment. In Traffic Control Centres a real hotchpotch of operator terminals are used. Usually exits a terminal per application. This is caused because all systems are proprietary systems and there is a big lack of openness. One of the main causes of this situation is the lack of a requirement for "functional layers" or "open interfaces" in a procurement process.

Important issues like maintainability and extensibility are poorly defined and the real meaning of such an approach is often noted too late; a new monolith is added to the infrastructure. Maintaining and extending such systems is not only complex, but often very expensive. Most developments depend on expensive proprietary libraries or even worse: recurring license fees.

A possible solution could be the definition and implementation of a fully open and free infrastructure for the development of control applications. However such a project would be too complex and first usable results would take too long. For this reason a smaller step was undertaken by us, with as prime objective the definition and implementation of an open-source "extensible graphic container" in short XGC. The project focuses mainly on the presentation layer and partly on the business layer of a system and provides the necessary libraries to develop specific dedicated systems

On the other hand, it is clear that XML has been recognized as an important technology and that first steps have been made for the transportation domain. However XML is only a basic framework for the definition of other XML languages. DTD's and XML-Schemas are a way to express data elements, their data type and in XML Schema also has basic functionality to describe data constraints. Replacing the actual exchange of data in an XML-based language has certainly many advantages, but the current developments in the XML-world show that it is also necessary to define the meaning or semantics of the exchanged elements. As far as we could find in our literature study, there are no serious attempts to define specific semantic web based ontology for transportation applications. A "common vocabulary" based on XML enables seamless

R. Tomás López V., Javier Samper J. and José Martínez. J. (2006).
INTEGRATING ONTOLOGIES AND USER INTERFACES - The XGC Approach.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 422-427
DOI: 10.5220/0001253204220427

integration of transportation systems and provides a solid foundation for future extensions.

This paper presents the work that we are currently developing to create a generic graphic container for transportation systems. The paper is organized as follow: Next section exposes the state of the art of road traffic vocabularies. Section 3 offers a first approach of a common vocabulary (ontology) for traffic systems. Section 4 gives the background information how we got here. Section 5 describes the XGC model and its functionality and finally the conclusions and next steps are exposed.

2 STATE OF THE ART

2.1 Existing Vocabularies

From several years ago, the use of mark-up languages has been established as a need to allow a reliable data exchange, emphasizing the use of standards like HTML first and XML afterwards. But nowadays we know that these architectures can be inefficient if the objective is more than a simple data exchange. However, it must be stated that the use of XML is still one of the most important tools in the information distribution area in Internet, since it has been used to define most of all the new languages that are used for data exchange in the Web.

There are currently vocabularies or languages that describe concepts and structures of data related to traffic, but they are only syntactic descriptions, lacking semantics. Therefore although there are many works and developments that use XML mark-up languages (information diffusion, data exchange, traffic modelling), no semantic specifications like the ones proposed in this paper are known.

For example, regarding road information diffusion by a management centre, there are some initiatives that make use of mark-up languages for information diffusion like CARS (Condition Acquisition and Reporting System) (Hamwi B and Choudhry O., 2005) in the United States, where XML and TMDD (Traffic Management Data Dictionary) are used as well as the project of the Research and Development Department of the Hokkaido Development Bureau within the ITS/Win Research programme where they propose a language that uses XML technology called RWML (Road Web Mark-up Language) in order to manage traffic information. This language consists of the description of a vocabulary that allows the information related to roads, weather forecasts, natural disasters, geographical regions, etc. to be represented. (Kajiya Y. et al., 2004).

For experiences of mark-up languages for road information using XML we underline: Traffic Data Mark-up Language (TDML), where XML Schema Data is defined and used instead of DTDs, and also in Europe TPEG (Transport Protocol Experts Group) with Road Traffic Message Application ML (Tpeg-rtml v0.4).

On the other hand, several specifications. have been developed about traffic modelling in the Transportation Research Center of the University of Florida: TMML (Traffic Model Markup Language). This markup language facilitates sharing data among the different software products for traffic modelling. They propose a specification that will cover all the data commonly used by a group of software products that manage information related to signalled intersections and roads. The use of TSDD (Traffic Software Data Dictionary) is good as a reference for the used vocabulary and labels used for identifying classes and attributes.

Other researches focus their interest on the development of specific vocabularies for accidents like CRML (Crash Records Mark-up Language), transport and logistics like or travel information like the one developed by Mitretek where the Society of Automotive Engineers (SAE) has developed an XML vocabulary for ATIS called Traveller Information Mark-up Language, based on the SAE ATIS data dictionary and the groups of standard messages (J2353 y J2354) defined using ASN. 1. The final result is a Standard mark-up language documented by XML Schema.

2.2 Why are Traffic Ontologies Necessary?

All the above mentioned demonstrates that the current computer traffic information processing is quite limited and can be improved from different points of view. These ideas are precisely the ones that have been taken as a starting point for this research.

The representation schema will have the following aspects:

- It will have to be interpreted by the computer and be easily exchangeable among applications.
- It will have to join the existing information representation standards in their syntactic aspects

As examples of possible applications based on the chosen representation schema, the following can be mentioned:

- An intelligent consultation and search system of traffic information.
- A tool for the visualization of the structured information.

If a user or an operator of the system needs to know specific information about the accidents occurred in a certain road, he will probably wants to know details about the vehicles involved, types of roads, etc, and therefore this type of question will not only involve one knowledge source but several. That is why the use of ontologies is so important, thanks to some of their characteristics like the distribution and the possibility of inferring non explicit knowledge beforehand.

As a particular case let us take an accident occurred in a particular toll motorway like "AP-7". If we use the terminological specifications of these types of roads and considering the attributes (number of toll sections, origin, destination, alternative roads, name) we could establish some questions that would immediately be solved by our inference system:

What number of toll sections are there in all the A-7 motorway? 4

What is the origin and destination for each one of these sections? Section 0: La Jonquera-Puzol, Section 1 Silla-Alicante.

3 OUR ROAD TRAFFIC DOMAIN

The transportation systems are composed by common objects such as roads, vehicles, etc. But the way of this objects are modeled is different depending on the systems (Van den Berg L., 2002). Therefore, the development of a generic user interface must contain a common representation of these objects.

This approach implies the definition and use a common ontology around the road traffic domain. This ontology is composed by several sub domains. Next, the traffic ontology is presented.

The road domain is also divided in several subdomains. The first describes topological features of the roads, and the second establishes a classification for the different types of them.

The topological subdomain is composed of the following objects:

- Roads: They represent a way between an origin and a destination. It is composed by a set of segments and links that belongs to the same road.
- Itineraries: They represent a way between an origin and a destination. It is composed by segments and links that belong to different roads.
- Segments: They represent one way road sections with the same number of lanes (i.e., a two way road section is represented as two one way road sections)
- Links: They represent the road network areas where two, or more, adjacent segments are connected.

In figure 1 we can see a partial view of the hierarchy of classes elaborated for the subdomain road classification

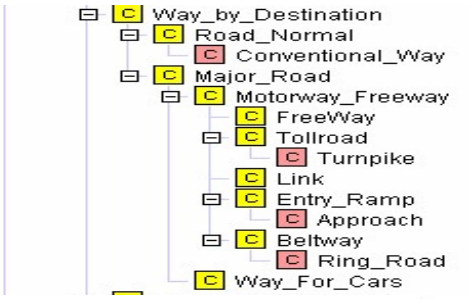

Figure 1: Partial view of the hierarchy of classes for the subdomain "Road Classification".

In this subdomain are identified different roads taking mainly into the count different criteria as roads by destination, and localization. By example we can classify the term "Street" under the criteria "Localization" with the following meaning: "They are the routes that are located within of a town, which means the space that includes buildings and in whose routes of entrance and exit they are placed, respectively, the signals of entrance and exit." The restrictions on the properties like "domain Public", "locatedIn Town" etc. will help in the automatic classification.

Next subdomain is devoted to the available equipment along the non urban network. The basic equipments that had been identified in this subdomain are:

- Data capture stations: These equipments are in charge of the transmission of road information. It can be subclassified in
  - Traffic data capture station: It provides data about the traffic behavior (flow, speed, density)
  - Meteorological station: It provides information about weather parameters.
- CCTV cameras: provides road images in real time that can be used to survey traffic status in concrete places.
- Emergency phones: These phones are directly connected to traffic control centers. They allow drivers to communicate directly with road operator.
- Variable Signals: A Variable Signal is a signal with the purpose of displaying messages that may be changed or switched on or off as it will be required.

The last subdomain presented of this traffic ontology defines the traffic behavior model. This subdomain describes the traffic behavior and its related parameters and it allow to share traffic information between systems. It is subclassified in two main groups: traffic parameters and weather parameters.

The traffic parameters are:

- Volume: the total number of vehicles on a section or lane during a given time interval (annual, daily or hourly periods).
- Flow rate (intensity): the rate at which vehicles pass on a section or a lane during a given time interval smaller than 1 hour (usually 15 m).
- Speed: It is a rate of motion expressed as distance per time unit.
- Density: the number of vehicles on a given length of a section or a lane in a particular instant
- Level of service (LOS): It defines the manoeuvre freedom of drivers in the traffic streams (Transportation Research Board, 2000). It is influenced by traffic and weather parameters.
-

The weather parameters affecting traffic behavior are:

- Visibility: It determines the recommended security distance between vehicles and the maximum speed drivers can drive.
- Road surface: the state of the surface defines the road vehicles adhesion
- Precipitations: Both, rain and snow have impacts on visibility and road surface status.
- Wind: Strong gusts can modify the trajectory of vehicles.

4 USER INTERFACE BACKGROUND

The constant increase of the technology and the need of more advanced systems to manage and control traffic imply the installation of more and more systems in the Traffic Control Centres. Currently, these systems are not integrated. Thus, the road operator has several interfaces, one for each system, to work with them.

To solve this problem the Transport Research Centre of the Dutch Ministry of Transport funded a research project "Paradigma" (Van den Berg L. and Klijnhout J., 2001) which main objective was to define and implement a generic infrastructure for traffic management applications.

Some recommendations from that project were:

- A modern traffic management system should be developed as a multi-tiered system with a clear separation between the persistency, the application and presentation layers.
- A loosely coupled asynchronous message interface is the preferred way of linking layers together. Such an interface, with the popular name Postman Pat can be build using the Java Message Service.
- The high value of eXtensible Markup Language (XML) for transportation was recognized (Tomás, Vicente R. et al., 2003). Besides the core-functionality in the basic XML-framework new "dialects" such as Resource Description Framework (for the semantics of traffic objects) and Scalable Vector Graphics (for the visual representation of traffic objects) were identified as important for the transportation domain;
- Scalable Vector Graphics (SVG) is a core technology for a generic user interface, because of its openness and the availability of a multitude of tools through different vendors.

Besides a multilayer approach, the use of XML and Java Message System JMS (Hunter J., 2001), the most important step in the development of the prototype was adding support for the processing of vector based graphics in the SVG format.

To enable SVG-support in the prototype an additional set of functions was added for viewing and manipulating SVG-documents. All specific 2D graphic support was replaced by SVG-support and functionality to interact with external systems was added in the beginning of 2004 the prototype was converted to its first official release and named "eXtensible Graphic Container" or XGC.

5 THE XGC ARCHITECTURE

The architecture of the present version of XGC is showed in figure 2

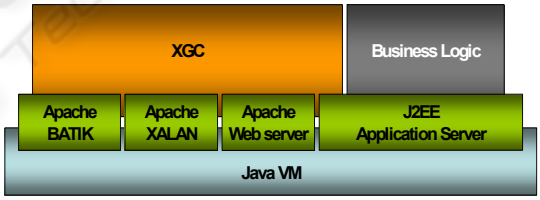

Figure 2: Architecture of XGC.

The XGC-library only works with Java version 1.4. Two additional libraries are a basic requirement for XGC to function: The Batik (http://xml.apache.org/batik/) and Xalan-J (http://xml.apache.org/xalan-j/) libraries from Apache.

To interact with its environment a Java Enterprise edition J2EE (Kay M., 2004) compliant application server is needed. The application server takes care of the message-based interaction with the "business logic" residing in the application server. It is the responsibility of the system developer to define, design and implement business logic in the application server.

In large scale applications where numerous maps and objects are used, the basic requirements are sufficient memory and a fast platform for the

Java Virtual Machine. A Pentium IV, PC-platform running at 2.5Ghz with a 512Mbyte memory can be defined as a minimum platform. The present version is tested on Linux and Windows. No difficulties are expected for other platforms.

5.1 XGC: How it Works

The architecture shows the two different part of XGC; the presentation and the interaction with the "business logic" through a message interface. We will focus first on the presentation.

To be able to show objects in their environment XGC can use "road maps" as a background. SVG is used to create these backgrounds, which can take the shape of a geographical correct map or a schematic view of the road network. Thanks to the support of bitmap formats, the background can also consist of a single bitmap file or a combination of a bitmap and vector graphics. The objects that compose "road maps" are defined in the topological road ontology described in Section 2.

The visual objects representing the actual traffic objects, such as emergency phones, variable message signs etc. are placed as a second layer on top of the background. These visual objects are identified in the equipment ontology. They are defined as re-usable SVG-symbols. As stated for the background, a bitmap image can also be declared as a reusable symbol.

The combination of a background map and the visual objects are declared in a SVG-file using the standardized XML-format. In principle this SVG-file can be created with standard SVG-editors. However, thanks to its XML-origin, the file can be dynamically created using Extensible Stylesheet Language (XSL) (Kajiya Y. et al., 2004).

At present the available version supports dynamic generation of the SVG-content.

For this process, two files need to be provided: An XML file containing the configuration and an XSL-file containing the transformation instructions. For the transformation process an Apache product, the Xalan-Java2 library is used. Xalan-Java 2 is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSLT Version 1.0 and XML XPath Version 1.0.

To create the XML and XSL-document it is recommended to use a commercial off-the-shelf XML-editor. XGC doesn't and will not include functionality to create the configuration files.

To visualize the SVG-document a software library is needed to place each element and possible attributes in a Document Object Model Tree and to render it on screen. This functionality is provided by means of another Apache product, the Batik library, which provides functions for loading, rendering and manipulating SVG-documents. XGC depends heavily on Batik, meaning all functional additions to Batik will also be available in XGC. At the moment XGC includes version 1.5.1 of Batik.

Thanks to Batik's use of the SVG-standard the elements of the SVG-Document can be altered programmatically by adding scripts to the documents or through its programming interface. In the case of XGC only the latter is used, dynamic scripting is not allowed.

However XGC is not a single Screen container; it runs as a Desktop application containing multiple internal frames. Each frame can be a Screen, which means that XGC is able to support multiple user screens at once. Although XGC is intended for SVG-based graphic rendering, other types of frames can be added. At the moment XGC supports screens for alarm- and plain text logging, HTML3.2 rendering (for instance for on-line help) and XML-viewing.

The configuration of the Desktop is placed in a single XML-file containing the application specific elements such as security issues, as well as the individual Screen elements and their attributes. Each Screen-element has attributes used to configure the Screen automatically at start up.

The configuration file is handled and managed by the applications' Configuration Handler, which reads the configuration file at start-up and places the configuration items in public variables, accessible to all classes in the application.

Each object can be made dynamic, meaning it can interact with the environment outside the platform it is running in. The simplest example is interaction with the operator. XGC supports standard "tool tips", meaning hovering over the object, a text can be shown. This is the default behavior included in SVG. On the other hand XGC contains support for clicking on the object and triggering object specific behavior.

As already mentioned, for this to happen, the "semantics" of the object has been identified. Clicking on the object results in reading the objects' semantic attribute, and based on its contents a context specific pop-up menu is activated.

Each menu item is defined in XGC's configuration file and it can contain text as well as graphic content. Each menu item trigger creates a command object, containing the identification of the selected visual object and the requested behavior.

This command object is send in serialized form via the message service to the application server

where the requested behavior is activated

Using this loosely coupled asynchronous message mechanism in stead of a standard closely coupled synchronous mechanism has different advantages:

- A request can take numerous sub-actions which takes a certain number of times. It is not desirable to have an operator wait until the system comes back with a response
- Message based interaction improves the systems' availability. Even when the application server is out-of-order, the message stays queued until the application server has regained service and is able to handle the request. The reply from the application server is also send via the message mechanism. This adds other important advantages to the ones already mentioned:
- It supports the "separation of concerns" paradigm. The business logic doesn't concern how the response will be rendered on screen
- By completely separating the layers by means of a message mechanism, the systems' scalability is improved substantially. It is really easy to add another listener to the response topic.

6 CONCLUSIONS

The developing of new systems, or upgrading others, can present problems if they have to work jointly and with the same objects but they do not share information. In these situations the definitions of commons vocabulary are a good option to share information and knowledge.

In this paper, the need to have a common vocabulary in the transportation has been identified and this road traffic ontology has been developed to develop traffic systems. This ontology has been used as a knowledge base to implement a generic user interface.

The generic interface presented has several good features: 1) based on a traffic ontology to develop the object systems and to provide semantic knowledge; 2) the container is developed in Java which allows to use it in different platforms; 3) open source approach.

The system has been tested in a real Dutch network. (figure 3). The traffic data flows have been simulated using data files containing real traffic information.

Currently the system is being used to develop the user interface of a new traffic system based on a Multiagent architecture and able to work with Traffic management plans.

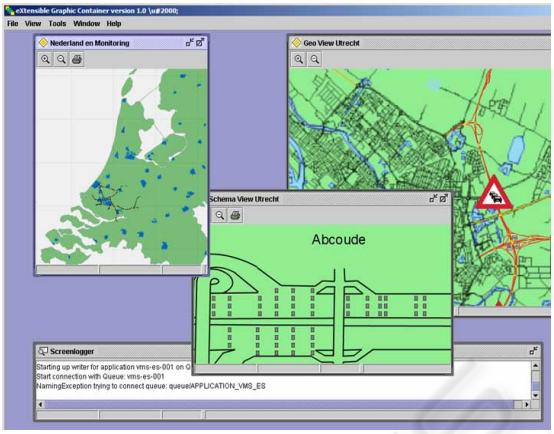

Figure 3: Dutch road network with XGC interface.

ACKNOWLEDGEMENTS

We would like to thank to Spanish Ministry of Science and Technology. This research has been supported by the CICYT project with reference TRA2004-06276 ("Development of a system using a conceptual infrastructure based on ontologies, to exchange good information between trucks and external entities ")

REFERENCES

Transportation Research Board "Highway Capacity Manual" 2000.

http://xml.apache.org/batik/

http://xml.apache.org/xalan-j/

Tomás, Vicente R. et al. New technologies to work with traffic management plans. p. 36-39. Traffic Technology International.. 2003

Hunter J. Java Servlet programming. 2nd edition. 2001

Van den Berg L. and Klijnhout J. Paradigma, From Open Systems Architecture to Open Systems. IX ITS World Congress Chicago 2001

Kay M. XPath 2.0. Programmers Reference. 2004

Van den Berg L. A basic need for effective traffic management. A universal operator interface. X ITS World Congress. Madrid 2002

http://www.w3.org/Style/XSL/

Hamwi B, Choudhry O. Condition Acquisition and Reporting System (CARS). ITS Canada Conference. 2005

Kajiya Y, Yamagiwa Y, Kudo Y, Road Web Markup Language-XML for Road-related Information Distribution. Transportation Research Board-Annual Meeting. 2004