

Internet Electronic Data Interchange with XML and JAVA

Karl Fürst

Institute of Flexible Automation, Vienna University of Technology, Vienna, Austria
kf@infa.tuwien.ac.at
<http://www.infa.tuwien.ac.at>

Thomas Schmidt

Institute of Flexible Automation, Vienna University of Technology, Vienna, Austria
ts@infa.tuwien.ac.at
<http://www.infa.tuwien.ac.at>

Abstract:

In this paper, a complete concept for Internet Electronic Data Interchange using XML is proposed and the realized prototype, which has been developed for an industrial partner to improve the current situation of parts delivery, is described.

Scope of the problem

The starting point of the project was a problem of our industrial partner (BMW Motors Steyr in Austria). During the process of parts delivery the company has to collect manually a lot of data either from the suppliers or from the carriers. Because this method of data collection is very time-consuming, it happens only if some problems during this process are becoming visible.

Because of the market pressure, which leads to reduced part stocks, and therefore to just-in-time production, it is necessary to increase the timely parts delivery and the integration between the enterprises. The system, which was developed with the industrial partner, fulfills two main functions:

The monitoring of the suppliers and the carriers for increasing the on-time departure performance of parts delivery.

The responsibility of parts delivery should be transferred to the suppliers. Therefore it is necessary to expose internal logistics data to the suppliers.

In both tasks there is need for a transparent, flexible and low-cost data exchange method, based on the latest Internet technology. The proposed system makes it possible that all relevant data for controlling the process of parts delivery are available automatically and electronically for the industrial partner, and all suppliers and carriers of this factory, at any time.

State of the art

Over the past several decades corporations have invested trillions of dollars in automating their internal processes. While this investment has yielded significant improvements in efficiency, that efficiency has been limited to internal processes. In effect, companies have created islands of automation, which are isolated from their vendors and customers. The interaction between companies and their trading partners remains slow and inefficient, because it is based on manual processes.

EDI (Electronic Data Interchange) has been heralded as the solution to this problem. EDI is defined as the exchange of data between heterogeneous systems to support transactions. This is not simply the exportation of data from one system to another, but the actual interaction between systems. Companies that have implemented EDI rave about the various benefits. In fact, these benefits can be expanded to a chain of suppliers.

There is a significant gap between the business benefits described above and the actual implementation of EDI. This is because the actual implementation of EDI is difficult and costly to implement. More importantly, however, it requires a unique solution for each pair of trading partners. Many people falsely proclaimed the Internet as the solution to this problem. By implementing EDI over a single network, our problems would be solved. Unfortunately, a network with a common protocol is still only a partial solution. This is because the systems implemented in each company are based on different platforms, applications, data formats, protocols, schemas, business rules and more. Simply “connecting” these systems over the Internet does not solve the problem.

Traditional EDI is based on fixed transaction sets. These transaction sets are defined by standards bodies such as the United Nations Standard Messages Directory for EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) , and the ANSI’s (American National Standards Institute) ASC (Accredited Standards Sommittee) X12 sub-group. Transaction sets define the fields, the order of these fields, and the length of the fields. Along with these transaction sets are business rules, which in the lexicon of the EDI folks are referred to as “implementation guidelines”.

The problems of traditional EDI are:

- Fixed transaction sets
- Slow standards evolution
- High fixed costs
- Fixed business rules

Realization of the proposed system

System-architecture

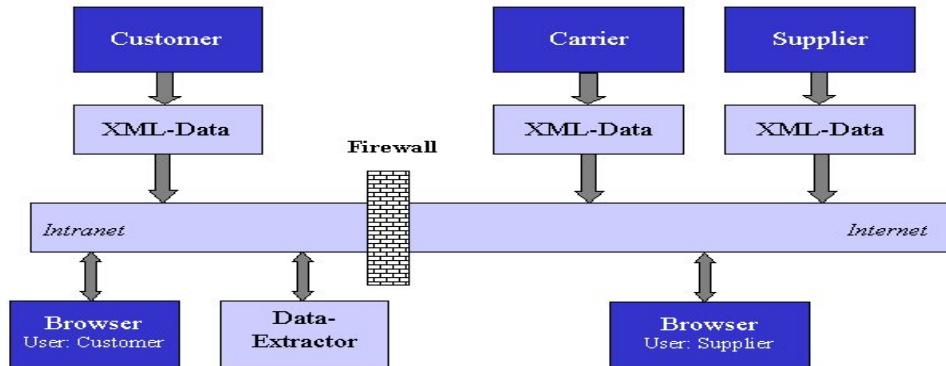


Figure 1. System-architecture
System-Architecture of the prototype

The system architecture in Figure 1 is based on the Internet technology and the new Internet language XML. Because the internal systems of the suppliers and carriers are not equal, it is necessary to extract the relevant data for the new system from these systems and convert the data locally into the international standardized format XML to make it available for the application, the so-called Data-Extractor, which runs on a Web server. The Data-Extractor is responsible for the whole data transmission between the different systems and should handle the requests from the different users working with a standard Web browser. The firewall between the Intranet and the global Internet is necessary to guarantee a high secure-level for the sensible industrial data.

The advantages of this system architecture are:

Platform independence of this system because the Data-Extractor is implemented with the programming language Java, exactly with Java-Servlets. The format XML is also platform independent.

The users can work with a standard Web browser like Microsoft Explorer or Netscape Navigator.

Using all advantages of the Internet technology like the availability of Internet or the low-cost data transmission over Internet.

The standard for Internet communication in the near future, XML (eXtensible Markup Language) , was standardized in the year 1998 by the World Wide Web Consortium and is a subset of SGML (Standard Generalized Markup Language) . SGML was standardized in the year 1986 by ISO (International Standards Organization) , but it is too complicated and therefore not used in a broad range. With the easier to use language XML it is also possible to separate data and markup. XML is a universal data format that allows computers to store and transfer data that can be understood by any other computer system. The advantages of XML are:

XML allows you to define your own tags and attributes.

XML is designed for describing structured data.

XML allows you to include metadata to describe your data.

If the metadata is included, XML supports the checking of data for structural validity (the so-called valid XML document).

At the moment there still exists no complete concept or realization of Internet-EDI using XML, because the XML standard is very new.

See , , and for more information about XML.

System-functionality

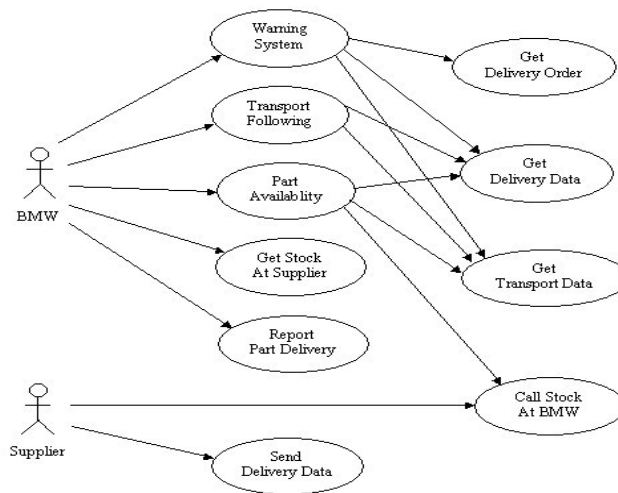


Figure 2. Use case diagram
Use Case Diagram

Using UML (Unified Modeling Language) the complete system model was designed. Figure 2 shows the simplified Use Case Diagram. In this diagram, the interactions between the following Use Cases are displayed:

Warning System: With this system the user can manage the whole part delivery, because the output of this Use Case is a table, where the delivery and transport data of all parts are controlled based on the delivery order. If the data differs, the system reports this via a red sign.

Transport Following: This Use Case reports the actual status of a part transport. The user gets all transport data from a part (e.g. time of delivery).

Part Availability: The user gets information about the actual part stock at BMW (the data come directly from the SAP-System) and also the delivery and transport data of a part as table with time of delivery and amount.

Calling Stock At Supplier: The user gets the actual stock at the supplier in form of a table with the part number, part name and amount.

Report Part Delivery: It is necessary to report the system the incoming of a delivery (e.g. to update the delivery and transport data).

Send Delivery Data: The supplier must send the actual delivery data to BMW.

Get Delivery Order: The actual delivery order of a part is calling from the SAP-system.

Get Delivery Data: Calling the actual delivery data of a part, which are stored locally in form of XML-files.

Get Transport Data: The actual transport data of a part is calling from the carrier.

Get Stock At BMW: The user gets the stock at BMW directly from the SAP-System.

System-implementation and -testing

The whole application (Data-Extractor) is implemented in Java. Platform independence is the main goal of Java and it has been achieved. Each Java program is both compiled and interpreted (see Figure 3). With a compiler, you translate a Java program (Program.java) on any platform into an intermediate language called Java bytecodes (Program.class) – the platform-independent codes interpreted by the Java interpreter (Java Virtual Machine). With an interpreter, each Java bytecode instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. So, Java bytecodes help make "write once, run anywhere" possible. See and for more information about Java.

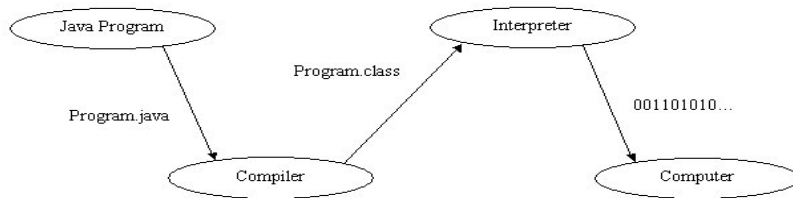


Figure 3. Java program

Each Java program is both compiled and interpreted

Because the application is running on a Web server, Java-Servlets, which realize the functionality of the Use Case Diagram (see Figure 2), are used. In general the rise of server-side Java applications is one of the latest and most exciting trends in Java programming. A Java-Servlet is a small, pluggable extension to a server that enhances the functionality of the server and runs inside a Java Virtual Machine on the server, so unlike applets, they do not require support for Java in the Web browser.

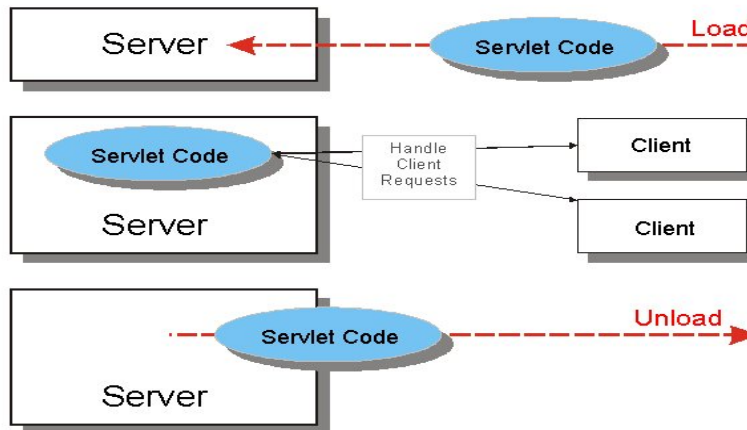


Figure 4. Java-Servlet
Life cycle of servlets

Each servlet has the same life cycle:

- A server loads and initializes the servlet

- The servlet handles zero or more client requests

- The server removes the servlet (some servers do this step only when they shut down)

See and for more information about Java-Servlets.

The Java-Servlets at the Web server handle the GET-Requests of the users (Customer, Suppliers) and work with the XML-Data. For these operations many software tools are developed since the standardization of XML. Under the developers are companies like JavaSoft, Microsoft, Oracle and IBM. This show how important the combination of Java/XML will be in the near future.

There are two major types of XML-APIs:

- A tree-based API compiles an XML document into an internal tree structure and allows an application to navigate that tree. The DOM (Document Object Model) is such an API, which is in the process of being standardized by W3C.

- An event-based API, on the other hand, reports parsing events (such as the start and end of elements) to an application through callbacks. The application implements handlers to deal with the various events. SAX (Simple API for XML) is such an API.

The servlets in the prototype use the DOM interface to operate with the XML data. After parsing the XML file (Figure 5), you can step through the internal tree structure (Figure 6) and save the necessary data into internal variables. For more information about Java/XML see and .

```
<SUPPLIER>
  <PART>
    <NUMBER>123456.00</NUMBER>
```

```

    ...
    </PART>
    ...
</SUPPLIER>

```

Figure 5. XML-file
Example of XML-File

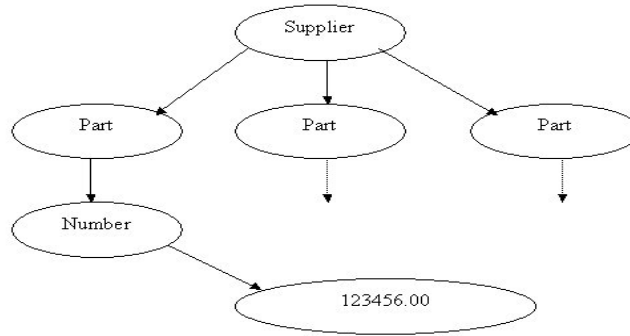


Figure 6. Document Object Model
Tree structure of DOM

After working with this internal data (e.g. calculate the Warning System under using a special logic), the servlet generate a HTML-Output and send this to the client. The user can see this output using a normal Web browser like Microsoft Explorer or Netscape Navigator.

The prototype is tested with simulation data, which are stored as XML files on the Web server. The connections to the Systems at BMW, the Carriers and the Suppliers are not realized now.

Conclusion

The results of testing the prototype give us the motivation to implement the next step of the project – the connections to the different systems. Temporary we will realize this integration with HTML-forms, which will be filled out manually by the suppliers and carriers. This method is an alternative way to become an EDI-partner, especially interesting for SMEs with a low data throughput. Only the interface to the internal SAP-System will shortly be realized to work automatically. So, the prototype can work with real production data from the different systems. After this real world test our industrial partner can better estimate the advantages and costs of such a system to decide the further steps for implementing the complete system.

Acknowledgments

The authors want to thank BMW Motors Steyr in Austria, especially Mr. Krebs and Mr. Brenner for their vision of an Internet supplier integration, their know-how and the general support of the project.

Bibliography

- [Beh99] Behme H., XML – Kunst der Stunde, IX 2/1999.
- [Cha98] Chang D., Harkey D., Client/Server Data Access with Java and XML, Wiley Computer Publishing, 1998.
- [Fla96] Flanagan D., Java in a nutshell, ÓReilly, 1996.
- [Gol99] Goldfarb C.F., Prescod P., XML-Handbuch, Prentice Hall, 1999.
- [Hel99] Universität Würzburg, Projekt Helios,
<http://www.wiinf.uni-wuerzburg.de/helios/startseite-main.htm> , 27.05.1999.
- [Hog98] Hogan M., XML and the Internet: Driving the future of EDI, POET-Software, 1998.
- [Hub96] Huber G., Java – Die Referenz, Heise, 1996.
- [Hun98] Hunter J., Java Servlet Programming, ÓReilly, 1998.
- [Joh99] John V., XML – Weltsprache für das Internet, OBJEKTSpektrum 5/99.
- [Mid99] Middendorf S., XML und Java, OBJEKTSpektrum 5/99.
- [Uni99] Universität Klagenfurt, Java-Tutorial,
<http://www.uni-klu.ac.at/unihome/header/tutorial/servlets/> , 10.09.1999.

Authors

Karl Fürst

University Assistant
Institute of Flexible Automation, Vienna University of Technology
Postal Address:
Gusshausstrasse 27-29/361
1040 Vienna
Austria
Telephon: +43-1-58801-36155
Fax: +43-1-58801-36199
E-mail: kf@infa.tuwien.ac.at
Web: www.infa.tuwien.ac.at

Karl Fürst – From 1989 to 1994: Studies at Vienna University of Technology, Department of Electrical Engineering. December 1994: Graduation in Industrial Electronics and Control Theory at Vienna University of Technology. Master Thesis: “Finite State Machine for Fuzzy Logic Based Local Collision Avoidance”. From 1995 to January 1999: PhD in Electrical Engineering at Vienna University of Technology. PhD Thesis: “Module Structure for Flexible Component Integration for High Working Speeds in Machine Vision”. Since February 1996: Research Assistant at Vienna University of Technology, Institute of Flexible Automation (INFA). Since July 1998: Head of the working group “Intelligent Product Design” at INFA. Since August 1998: Teaching Assistant at INFA. Currently working on: Virtual/Extended Enterprise, Intelligent Product Design, Dynamic Development Tools, Optimal Enterprise Operations Point. Karl Fürst has organized, chaired and spoken at the INFA Product Data Management Conference in May 1999 and has spoken at the following industry conferences: IECON98, CPÍ99.

Thomas Schmidt

Research Assistant
Institute of Flexible Automation, Vienna University of Technology
Postal Address:
Gusshausstrasse 27-29/361
1040 Vienna
Austria
Telephon: +43-1-58801-36157
Fax: +43-1-58801-36199
E-mail: ts@infa.tuwien.ac.at
Web: www.infa.tuwien.ac.at

Thomas Schmidt – From 1993 to 1999: Studies at Vienna University of Technology, Faculty of Electrical Engineering and Information Technology. November 1999: Graduation in Computer Technology at Vienna University of Technology. Master Thesis: “Internet Electronic Data Interchange for Flexible Logistics at a Motors-factory”. Since December 1999: Research Assistant at the Institute of Flexible Automation, Vienna University of Technology. Currently working on: Virtual/Extended Enterprise, Java, XML. Thomas Schmidt has spoken at the following industry conference: CPÍ99.