UNIVERSITY OF CRAIOVA
FACULTY OF MECHANICS
SMAT 2008

SECOND INTERNATIONAL CONGRESS
AUTOMOTIVE, SAFETY AND ENVIRONMENT
23 - 25 October, 2008 – Craiova, Romania

CAD-PLM INTEGRATION USING THE WEBDAV PROTOCOL

Dinu COVACIU, Ion PREDA, Vasile CÂMPIAN, Ovidiu-Vasile CÂMPIAN

Abstract: *Product Lifecycle Management or PLM is a term used for managing the entire product lifecycle, from its conception, through design and manufacture to service and disposal/recycling. A mechanical CAD and PLM integration enables the PLM system to manage CAD design objects (assemblies, parts, drawings, etc.), to control the access to them, to track changes and to maintain the appropriate relationships between the CAD objects and the PLM's product structure. WebDAV is a protocol dedicated to versioning and authoring, a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers. This paper presents a CAD-PLM integration method using this protocol, and a tool developed based on this method.*

Keywords: *PLM, CAD, WebDAV, DASL, versioning.*

1. INTRODUCTION

PLM is a wide research issue exploring the information and knowledge in the entire product lifecycle to improve and increase innovation for a given product. The first attempts to define Product Lifecycle Management (PLM) was in the late 80s. Since then, the PLM market has grown. The whole life cycle of a product is usually very long, and this causes a disagreement regarding the PLM definition, its stages and the tools it comprises. It also makes it difficult for a single tool to include all the capabilities required, which in turn causes a lack of integration among several tools used throughout the PLM stages. Therefore, the product's knowledge, which most of the time comes in disparate formats, is not always shared among its life-cycle. For instance, there is no standard regarding Computer Aided Design (CAD) formats, and the technology to migrate from one format to another only handles geometry representations of the product (Bill Of Materials are being included already, but there is no standard either).

Diverse CAD/CAM systems are adopted by manufacturing industry, and the need for interoperability and data exchange between different engineering platforms has considerably increased. Effective and efficient data exchange is critical to effective engineering collaboration, which is of significant importance to successful implementation of Product Lifecycle Management (PLM) systems. Ideally, once CAD data is created, it could be reused again and again without any additional labor. CAD file should be transferred seamlessly back and forth between different CAD systems, up and down inside the manufacturing organization.

2. THE COMPONENTS OF A PLM SYSTEM

A PLM system is not a punctual solution, like PDM, CAD, CAD, CAE, Office. A PLM System implies two functions:

- efficient management of the corporate intellectual capital – facilitate the accuracy, integrity and security of all data;
- efficient use of the corporate intellectual capital – availability of all corporate information in the right place and format for the right users (people and programs).

These functions are both generic and does not establish a way for processing the content of the corporate intellectual capital. The function of processing the content is realized by the authoring applications, like CAD, CAE, Office etc. The separation of management and creation functions is important for defining the PLM goal, and for his understanding.

For any PLM system some components are required:

- an application server, where also a database management system may be installed;
- a protocol of communication between the clients and the server;
- client software for implementation of the protocols (a library of functions);
- an application integrated in the user's CAD system, which will use the above library, written according to the CAD system API;
- an interface for accessing the database on the server; this may be included in a specialized application, or may be web based (using a regular browser like *Internet Explorer* or *Firefox*).

3. CAD-PLM INTEGRATION

Integration of a mechanical CAD system in a PLM solution allows the management of CAD objects (assemblies, parts, drawings), changes and relationships between CAD objects and the PLM product structure.

The key benefits of an integrated CAD/PLM systems are:

- secured access to the CAD objects;
- globally distributed access to a single source of CAD objects;
- a better management of changes (explicit tracking of new versions and revisions);
- a better reuse of common parts and assemblies.

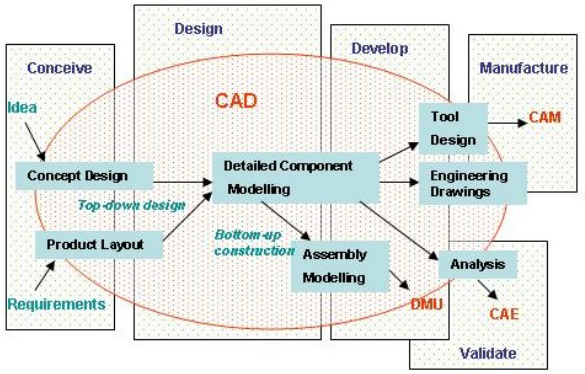

Fig. 1. CAD and the product lifecycle

The characteristics of an integrated CAD/PLM system include:

- document management;
- access control;
- product structure;
- project status;
- attribute management;
- change management.

A key feature of the integration is the version control, which facilitate the change management. The version control is the ability of uploading a resource in a version control system, where this resource has multiple revisions stored, and the revisions may have multiple relationships with their predecessors and successors. The server maintains these relationships, reports the revision history and controls the write permissions for these revisions using the *check-in* and *check-out* operations. The parallel development allow a better availability of resources in a multi-user environment. More users may *check-out* the same revision of a resource, the check if they have the same revision and finally merge the changes. The configuration management means to bring together consistent revisions of a resource.

4. THE WEBDAV PROTOCOL

The key element for implementing a version control system is the communication protocol. A protocol dedicated to versioning and

authoring is WebDAV (*Web-based Distributed Authoring and Versioning*), a set of extensions to the HTTP protocol. WebDAV allows the users to collaborative edit and manage the files on a remote server.

In figure 2 is an example of client/server communication for updating a document using a versioning protocol (WebDAV).

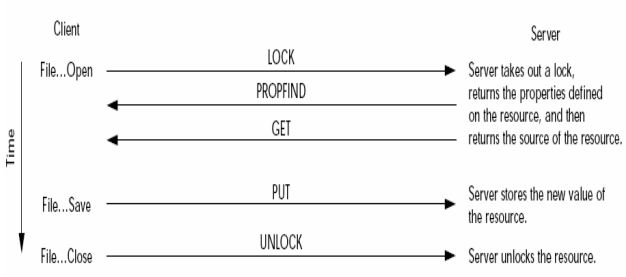

Fig. 2. Example for checking-out a resource using the WebDAV protocol

WebDAV is a standard under development; the basic document is a RFC (*Request for Call*) [3]. In a first phase, the working group has defined a set of HTTP *(Hypertext Transfer Protocol)* extensions for six new capabilities. These are: overwrite prevention, properties (add/ query/ edit), namespace management, version management, advanced collection management, access control.

The work of the WebDAV group is focused on three documents:

- a scenario, containing a set of short descriptions for using the distributed authoring and versioning capabilities; the name of the document is *"HTTP-based Distributed Content Editing Scenarios"*, http://www.ics.uci.edu/pub/ietf/webdav/ scenarios/draft-ietf-webdav-scenarios-00.txt;
- a requirement document, describing the functional requirements at high level for authoring and versioning management; the name of the document is *"Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web"*, ftp://ftp.isi.edu/ in-notes/rfc2291.txt;
- a protocol specification, describing the new HTTP methods, headers, request body and response body, for implementing the requirements for authoring and versioning; the name of the document is *"Extensions for Distributed Authoring on the World Wide Web - WebDAV"*, http://www.ietf. org/internet-drafts/draft-ietf-webdav-protocol-08.txt.

Figure 2 illustrates the working way in a WebDAV session. The arrows indicates the information flow associated with a request. All the requests use the WebDAV protocol plus HTTP 1.1. The user of an application that is WebDAV compliant select the resource to be edited using a standard file open dialog. Then the application uses the *LOCK* method to block the resource (file), the *PROPFIND* method to read the resource properties, and then the *GET* method (HTTP) to download the resource content – the file. After editing the resource, it is used the *PUT* method (HTTP) to upload the resource back to the server, and the *UNLOCK* method (WebDAV) to unblock the resource; the other users will be able to access it again.

WebDAV add seven new methods to the HTTP protocol: *GET, HEAD, POST, OPTIONS, PUT, DELETE, TRACE*. That methods are use for overwrite prevention (*LOCK, UNLOCK*), metadata management (*PROPFIND, PROPPATCH*) and namespace management (*COPY, MOVE, MKCOL*).

The WebDAV properties (metadata) are name-value pairs, where the name is an URL (*Uniform Resource Locator*) and the value is an XML (*eXtensible Markup Language*) sequence.

As the WebDAV progressed, it was clear that the version control mechanism is very important and difficult to manage, and therefore a special attention is needed. **DeltaV** is an extension of the WebDAV protocol. DeltaV is picking up where WebDAV left off, extending the protocol with versioning and configuration management support.

On the basis of HTTP and WebDAV, DeltaV add eleven new methods. The methods *VERSION-CONTROL, REPORT, CHECKIN, CHECKOUT* and *UNCHECKOUT* provide the versioning capability.

Very important are the *CHECKIN* and *CHECKOUT* methods, used to check-in a

resource on the server and to check-out a resource from the server, respectively. The check-in action automatically creates a new version; the existing version becomes predecessor of the new version. A request/response example at the execution of *CHECKOUT* method is shown in figure 3.

```
>>REQUEST

  CHECKOUT /foo.html HTTP/1.1
  Host: www.webdav.org

 >>RESPONSE

  HTTP/1.1 409 Conflict
  Content-Type: text/xml; charset="utf-8"
  Content-Length: xxxx

  <?xml version="1.0" encoding="utf-8" ?>
  <D:error xmlns:D="DAV:">
    <D:must-be-checked-in/>
  </D:error>
```

Fig. 3. *CHECKOUT* request/response example

The header is HTTP type and the request body is XML formatted. The syntax is similar for all WebDAV and DeltaV methods.

The other methods defined by DeltaV are: *MERGE, MKACTIVITY, MKWORKSPACE* and *BASELINE-CONTROL*.

DASL means *DAV Searching and Locating*, and is another WebDAV extension. Like DeltaV, DASL add new methods. The DASL working group is no more active, but his activity was continued and the term DASL was replaced in the last months with "*WebDAV SEARCH*"; however, the authors of the latest documents consider that these two terms are interchangeable, and refers to the same extension of WebDAV. The last update of the *WebDAV Search* specification was done in the summer of 2008, and the document is valid until January 2009 (the document is an *Internet Draft,* and expires in six months), but this not means that the information include will be no more valid.

The main contribution of DASL is the introduction of the method *SEARCH*. The client invoke the method to initiate a server-side search. The query is defined in the request body. The server should issue an entity corresponding to the WebDAV format.

5. TOOLS AND APPLICATIONS

Neon is a client library that implements the HTTP and WebDAV protocols, written in C language. The features include:

- high-level wrappers for common HTTP and WebDAV operations (*GET, MOVE, DELETE,* etc.);
- Low-level interface to the HTTP request/response engine, allowing the use of arbitrary HTTP methods, headers, etc;
- Authentication support including Basic and Digest support;
- SSL/TLS support, exposing an abstraction layer for verifying server certificates, handling client certificates, and examining certificate properties;
- Abstract interface to parsing XML, and wrappers for simplifying handling XML HTTP response bodies;
- WebDAV metadata support; wrappers for PROPFIND and PROPPATCH to simplify property manipulation.

Neon is a free, *open-source*, software library (author Joe Orton [9]), distributed under *GNU Library GPL* license.

It is important to mention that Neon does not implements the DeltaV and DASL methods.

In order to use the *Neon* also for versioning and searching, some new function were added to the library. These functions implement the main methods of DASL and DeltaV (SEARCH, *REPORT*, *VERSION-CONTROL, CHECKIN, CHECKOUT*, *UNCHECKOUT*).

All the important CAD systems had application programming interfaces (API) based on C++. Some examples: Catia V5 – CAA; Pro/Engineer – Pro/Toolkit; Unigraphics – UgOpen; I-deas – Open I-deas; NX (after integration Unigrahics/I-deas) – NX Open; Inventor – Open Inventor; AutoCAD - Object ARX; SolidWorks – SolidWorks API.

Because C++ is generally used for CAD programming, the first step for writing a CAD/WebDAV client was to write a C++ wrapper of the *Neon* C library (*NeonPlus*). This is in fact a C++ object, with his constructor and methods – a class called *WebdavDlg.* This class

is called from applications written for each CAD system. The general layout of such application is in the figure 4.

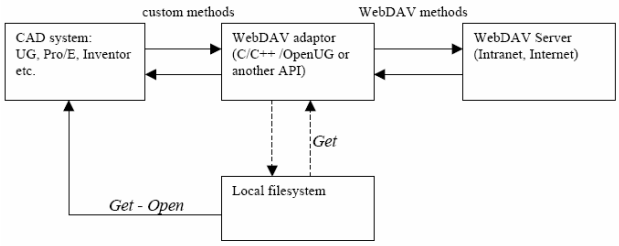

Fig. 4. Basis layout for a CAD-PLM integration based on the WebDAV protocol

The call of the NeonPlus functions is simple; a sample is shown in figure 5, for creating a new collection - the WebDAV method *MKCOL*.

```
.................
WebdavDlg *dlg;
dlg = new WebdavDlg(user_name, user_pwd,
     server_path);

.................

dlg->Mkcol(coll);
delete dlg;
.................
```

Fig. 5. Creation of a collection using the NeonPlus class WebdavDlg

The main methods implemented in the CAD system are described in the table 1. The user interface is made using the menu structure of the host application (CAD), and the dialog mechanism from the corresponding API.

Client	Server
Get Part – request the current version of a part or assembly	- Identify the current version - Check if the resource (file) is not locked - Lock the resource - Send the file to the client
Put Part – check-in a modified part, as a new version	- Save the file as a new version - Unlock the file
Get/Put Properties – add custom data to a part (according to internal norms)	- For adding/editing a property, lock the file before and unlock it after setting the prop.
Get Previous Version – request a previous version of a part (assembly) and download (check-out) the file from the server	- Generate a report (list of all saved versions for that file) - Identify the requested version in the collection called *history* (a folder containing the version history) - Send the file to the client, like for the *Get Part* request.
Search for Parts – search for parts on the server, using *metadata* (properties) based queries	- Execute the *Search* method - Generate a list of files (parts) that match the request.

Table 1. Examples of the implemented methods

Figure 6 show an example dialog for an implementation of the version control in a CAD system.

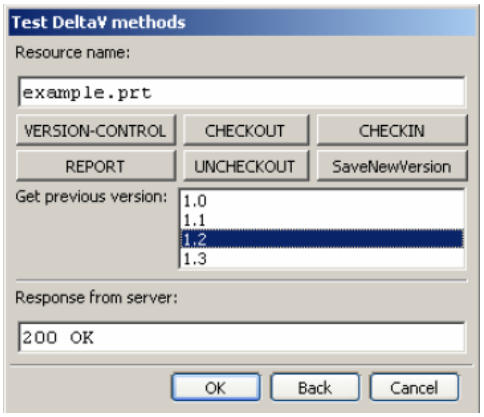

Fig. 6. Example dialog for version control implemented in a CAD system (Unigraphics)

6. CONCLUSION

Today, many companies are involved in the processes of the automotive industry. The OEM companies, subcontractors and providers, should all collaborate and access the same data. There are specialized applications used in automotive design. These applications use proprietary data formats and don't facilitate collaborative engineering. The collaborative methods are limited to choosing solutions, merging subassemblies and copy/add data.

The commercial CAD systems (from AutoCAD to Pro/Engineer, NX and Catia) don't offer integrated functions or interfaces for

authoring and versioning. For that, the regular open/save functions should be changed by adding the features of check-in/check-out on a WebDAV server. A WebDAV system is composed by a server and a client application that "speak" the WebDAV protocol.

It is not easy to develop a solution independent by the CAD system. Instead, it is possible to develop a library with functions that implement the WebDAV methods, then use the CAD system API to write the user interface and communication functions. Many WebDAV client open-source applications are written in Java, but this language is not supported by most of the CAD systems. The most important CAD systems include API based on C++, and this seems to be the most appropriate programming language. There are no open-source WebDAV clients written in C++, but there are some C implementations, and it was possible to complete a C library and write a C++ wrapper.

So, the basic elements for implementing a version control system in a CAD system are the WebDAV protocol, a WebDAV server, an open-source C library, a C++ extension of this library and custom commands defined in the API provided by each CAD system.

The integration of WebDAV in the CAD systems is a solution for updating the traditional and individual activity, facilitating a much collaborative work.

7. REFERENCES

[1] John Stark, *Product Lifecycle Management - 21st century Paradigm for Product Realization,* Springer 2004, ISBN 1-85233-810-5.

[2] Sidney Hill, *A winning strategy,* Manufacturing Business Technology, Dec. 2006.

[3] IETF, *RFC2518: HTTP Extensions for Distributed Authoring and Versioning – WEBDAV*, http://www.webdav.org, February 1999.

[4] Jim Whitehead, *DeltaV: Adding Versioning to the Web*, http://www.webdav.org/deltav/www10/deltav-intro.htm, 2001.

[5] James J. Hunt, Jurgen Reuter, *Using the Web for Document Versioning: An Implementation Report for DeltaV*, http://www.ipd.uka.de/~reuter/publications/ deltav.pdf , 2000.

[6] E. James Whitehead, jr., Meredith Wiggins, *WEBDAV: IETF Standard for Collaborative Authoring on the Web*, IEEE INTERNET COMPUTING, http://computer.org/internet/ September - October 1998.

[7] Dinu Covaciu, Horia Brădău, Ion Preda, *CAD-WEBDAV Adaptor – Premise for a light PLM solution*, CONAT20046005, Braşov, 2004.

[8] Dinu Covaciu, Ion Preda, Vasile Câmpian, *Implementation of a Versioning System in a Commercial CAD System*, AMMA2007, Cluj-Napoca, 2007.

[9] Joe Orton, *The Neon library*, http://www.webdav.org/neon, 2001-2008.

[10] Alan Johnson, *Understanding Product Lifecycle Management*, Manufacturers Monthly Australia, May 2007.

Dinu Covaciu, eng., Ph.D. student, system engineer at Transilvania University of Braşov, Automotive Department; email: dinu.covaciu@unitbv.ro

Ion Preda, professor, eng., Ph.D., Transilvania University of Braşov, Automotive Department; email: pion@unitbv.ro

Vasile Câmpian, professor, eng., Ph.D., Transilvania University of Braşov, Automotive Department

Ovidiu-Vasile Câmpian, professor, eng., Ph.D., Transilvania University of Braşov, Automotive Department; email: c.ovidiu@unitbv.ro