A VIRTUAL PROJECT HOUSE INFRASTRUCTURE FOR DISTRIBUTED DEVELOPMENT PROCESSESⁱ

Gerd Frick, Klaus D. Müller-Glaser
Forschungszentrum Informatik (FZI), Karlsruhe, esm@fzi.de
GERMANY

Many manufacturer-supplier relationships require intensive collaboration between engineering teams of several companies throughout the development process. The infrastructure presented in this paper follows the vision of bringing these teams together virtually under one roof (called a 'Virtual Project House').

Addressing the problems of manual information exchange and unorganized data redundancy in distributed development processes, our Internet-based infrastructure concentrates on sharing of data (e.g. engineering artifacts) and realizing data-centric workflows across site and company boundaries. The software interface between the partners is platform and tool-vendor independent, based solely on agreed data models and collaboration protocols and implemented with standard Internet technology (XML, HTTP).

1. INTRODUCTION

Complex and innovative technical products are rarely being developed entirely by a single company. Instead, a manufacturer 'outsources' the development (and production) of components or subsystems to suppliers with specialized know-how whereby teams of either company have to collaborate intensively throughout the development phases.

Bringing engineering teams together under one roof (in analogy to "just-in-sequence" production concepts with manufacturer and suppliers sharing the same assembly line on one site) would provide a nearly ideal environment. Indeed this concept has been put into practice in pilot projects in automotive industry – at the cost of establishing shared localities and moving teams from their home company site to the building shared with their project partner (called 'design house' or 'project house'). Our vision is to do this *virtually* by a flexible Internet-based infrastructure called *Virtual Project House*, established for a limited time as needed and leaving everybody at their home site (Frick et al., 1999).

System development processes are strongly related to the products, the engineering artifacts generated and worked on during the various development phases (for electronic systems e.g. requirements and specification documents, system models, software code, verification and test data, manuals etc.). The same

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35399-9_52](https://doi.org/10.1007/978-0-387-35399-9_52)

principle applies to micro-processes (workflows) such as a change request passed around for review. *Distributed* development processes – crossing site and company boundaries – therefore can easily suffer from ineffective handling of shared data. Manual information exchange procedures (including printing and mailing of documents, shipping of diskettes or CDs, traveling to meetings etc.) are time-consuming and may lead to delayed synchronization of work (the results are interrupts in the individual design tasks or additional iterative changes due to intermediate work with outdated information). Furthermore, unorganized redundancy in project data management is a permanent source of inconsistencies.

Our Virtual Project House approach (trying to eliminate major drawbacks of distributed development processes by imitating means that would easily be available in a shared local environment) therefore concentrates on management and sharing of data in distributed engineering projects by means of a common virtual repository (chapter 2) and the realization of workflow mechanisms on this foundation (chapter 3). Chapter 4 describes the client application, the Project Browser. This is only a basic infrastructure and may be complemented by e.g. teleconferencing facilities for virtual meetings (although occasional face-to-face meetings surely cannot be replaced altogether).

2. VIRTUAL PROJECT DATABASES

The situation to be tackled is two or more teams of engineers from different organizations at different locations working collaboratively on partially shared data. As the core of our Virtual Project House infrastructure we elaborate the *Virtual Project Database* approach proposed in Frick et al. (2000) for solving a data sharing problem of this kind (see Figure 1). It is able to handle data distributed physically over several sites and organizations and stored by different data management tools in a number of repositories at each location (e.g. software configuration management systems, document or content management systems, or relational databases).

Figure 1 – Virtual Project Database (overall architecture)

It respects the facts that project data are heterogeneous in nature, that different tools are best-suited for different kinds of data and that different companies use different tools for the same type of data.

2.1 Architecture

As shown in Figure 1, the Virtual Project Database is realized in a multi-tier client/server architecture by a group of coupled *Project Servers*. Each of the servers represents a set of local repositories; the group represents the virtual overall database for the entire project (physically composed of parts residing in the distributed repositories). Clients only contact the local server that behaves like a 'proxy' for the external servers. Based on an address space of global identifiers, client requests are routed to the (local or remote) repository where the original object or a replica is stored.

Figure 2 shows the internal Project Server architecture. The core application logic is encapsulated by the Enterprise JavaBeans™ (Sun/EJB, 2000) based *Application Server*ⁱⁱ. It is directly connected to by the standalone clients (a Java™ application; Gosling et al., 1996). The client functionality is also provided by the *Web Presentation Server* that realizes an HTML based user interface for access via a standard WWW browser. It has been implemented with JavaServer Pages™ (Sun/JSP, 2000) technology. The purpose of the *Integration Server* is the communication between the different project servers. It is contacted by remote *Integration Clients* that are part of the *Virtual Database Kernel*.

Figure 2 – Project Server

The Virtual Database (VDB) Kernel is the most vital part of the Application Server. It is responsible for uniform retrieval and storing of persistent data objects as instances of the integrated data model classes described in section 2.2. Internally it

has a two-layer architecture (see Figure 3). A general *Data Service* component is the single access point to the 'global' information space and routes the requests to individual *Data Manager* components that are adapted to and communicate with the underlying data management tools. Data Managers have been implemented for accessing objects in relational databases, XML files (Bray et al., 1998), LDAP directories (Yeong et al., 1995) and, as a representative of special-purpose repositories, the Hyperwave Information Server™ (Hyperwave, 2000) (via its proprietary client/server protocol).

Figure 3 – Virtual Database Kernel

One additional Data Manager is the already mentioned Integration Client. Seamlessly integrated into the VDB Kernel it is responsible for providing access to remote data objects via external project servers (Integration Servers). Integration Client and Server communicate with an application-level client/server protocol using XML for message marshalling and HTTP as the transfer protocol. It is fully session and transaction based, so remote objects can be kept in a local cache until transaction commit for reducing network traffic.

2.2 Integrated Object-Oriented Data Model

The logical backbone of the integration of heterogeneous and distributed data sources towards a virtual overall database is an object-oriented, integrated and extensible data model. The base object class first of all provides for object identity. A global object ID is composed of an organization's identifier, a repository identifier and the local object ID within a repository. An inheritance tree of various object classes is growing below the base class. Major classifications are derived from the different kinds of data and repository systems and can be refined to different levels of detail.

The main object model is complemented by a reference model, capable of linking objects in different repositories. More specifically, we defined references to link only 'documents', whereby *Document* is a generic class including product and workflow data, but excluding administrative objects such as users or roles. The links are kept separately ('out-of-line'), i.e. there is no need of altering the individual data models of the various repository tools. There is no requirement for a central link

database, either. Link management is decentralized by separating references from reference anchors and treating these link objects like all other objects with normal VDB mechanisms. The local link objects are stored in a general-purpose relational database (which supports the virtual project house concept in general since it is also used for administration and workflow data; see chapter 3). The references can be given different types for specifying special semantics.

Beyond the explicit document reference concept there are implicit associations between individual object classes (e.g. linking an object to a user as its owner), that normally are implemented 'inline' as intrinsic parts of the data objects and backed by individual repository concepts.

2.3 Heterogeneous and Distributed Persistency

When used by the application a persistent object is retrieved from and stored back to its permanent storage location, which is determined by its unique global identifier. This process is transparent to the application. The Java classes that implement the data model contain class-specific functionality for data input/output against the respective native repository. (E.g. object contents managed in a relational database is accessed by means of JDBC/SQL code.) There may even be multiple I/O implementations for different repository (persistency) types. These class and persistency type specific routines are used by and complement the general retrieval and transaction functionality implemented by the respective Data Managers.

All classes additionally implement XML input/output. Thus any object (in main memory) can be serialized to or initialized from an XML fragment (containing its property values and descriptors of links to other objects). This is used for remote data transfer via the Integration Server. XML serialized data objects or object deltas (i.e. the changes made to an object) are embedded in the client/server protocol messages sent between the Project Servers. XML serialization can also be used for data export to an XML file.

Please note that at the interface between two organizations (i.e. between two Project Servers) the data model has only to be agreed on at the XML level. The bindings to native persistent storage formats for the object classes can be different on both sides. The XML interchange format for an object class can be defined using the standard Document Type Definition (DTD) concept (Bray et al., 1998) or the more powerful XML Schema language (Fallside, 2000).

Ideally the native storage format is already XML. In many domains there are industry-wide standardization efforts for XML based document and data exchange formats, e.g. the MSR consortium in the European automotive industry (Perenthaler et al., 1998) or the Object Management Group's XMI framework for metadata interchange (IBM, 2000), used e.g. for UML models.

For agreed-on binary formats (e.g. file formats of engineering or office tools used on both sides) an alternative to the full-XML approach is possible: serializing only the object meta-data in XML and embedding the content as encoded binary (Fallside, 2000). This harmonizes with the upload/download approach described in chapter 4.

2.4 Access Control

There is a further difficulty in realizing the Virtual Database Kernel: the variety of access control policies enforced by the different underlying data management tools. While authorization can be handled by the Data Managers (wrapping the access control mechanisms performed by the repository tool) once the user is authenticated, user identification (authentication) requires credentials provided individually by the user.

Ideally the tools are able to share user information based on a system-wide directory (e.g. LDAP based). Thus a 'single sign-on' approach is possible, where the user logs in once and automatically is authenticated to all tools sharing the same authentication base. Some care must be taken for remote sessions at external Project Servers. Remote logins may be enabled by a limited trust relationship between different directory servers. If tools are not aware of LDAP based authentication, the VDB may implement single sign-on by having its own user management and collecting user account information for the various tools in its user records.

Please note that the access control mechanisms restrict the global information space to individual views according to the access rights of an authenticated user. As far as allowed by the underlying data management tools the user context can be augmented by role information and access control can be based on user/role/group based ACLs (access control lists). This leads to more elaborated view concepts (implemented in the workflow subsystem, see below).

3. WORKFLOW MECHANISMS FOR LOCAL AND DISTRIBUTED MICRO-PROCESSES

Besides surmounting organizational boundaries for coordinated handling of shared data the Virtual Project House concept aims at efficient information flows between distributed developers. Together with partners from automotive industry companies some practical scenarios were elaborated. They are related to review and delegation processes in requirements and change request management and organizing (e.g. review) meetings. A detailed description of the scenarios is beyond the scope of this paper. They can essentially be reduced to certain kinds of small-size documents (workflow documents) being passed around for processing by users acting in roles. The workflow documents generally are linked to some other documents, which may be engineering artifacts or previous workflow documents.

Workflow support therefore can be reduced to a general message delivery concept, using special kinds of messages and binding mailboxes to roles. The new object classes (including different types of messages and other workflow document classes, adapted to the needs of the scenarios being implemented) extend the core data model as *Document* subclasses. The reference concept can be used for linking between the documents (cf. section 2.2).

In our prototype implementation we chose a simple but full-custom implementation of the messaging layer on top of a relational database system. Thus we were able to realize a role-based messaging/workflow concept and user/role/group-based access control (resp. views) in full flexibility. The message delivery and workflow functionality is implemented as part of the application logic

(Figure 4, cf. Figure 2). It is reduced to data manipulation applied to special classes of data objects stored in the relational database. Message delivery beyond organizational boundaries leverages the normal remote data manipulation mechanisms.

Figure 4 – Refined Application Server

In production environments an existing mailbox system (e.g. Lotus Notes®, Microsoft® Exchange, IMAP or NNTP based) could be used as the messaging/workflow repository, with or without utilizing its message delivery mechanisms. In the latter case remote delivery can be routed through the project server (using special mailboxes observed by client agents) or via standard Internet delivery mechanisms (e-mail or Usenet). Special care has to be taken for preserving object identity: A received message in the recipient's mailbox must be identifiable as a replica of the sent message kept in the sender's archive mailbox so that references to such messages can clearly be resolved by both sender and recipient side systems.

4. PROJECT BROWSER

The *Project Browser* is our client application that presents the Virtual Database and workflow functionality to the user. Large parts of the functionality have been implemented as both a Java application and a JSP based Web interface (HTML/JavaScript).

Aware of the object-oriented data model the Project Browser first of all allows the user to seamlessly navigate through the distributed project information space (constrained by visibility based on access control). The navigation paths are determined by object associations and document references. Furthermore it is able to launch viewers/editors for the most important object classes (either built-in or external) and provides functionality for an upload/download interface for editing complex data as files with dedicated development tools. Finally, there are user dialogs for the workflows defined by the scenarios.

The Web interface in principle enables access to a Project Server even from a remote site. Thus there is the option of collaborating with external partners using one Project Server only. This is a light-weight Virtual Project House where one partner provides the entire infrastructure and stores all data and the other partner (e.g. a single developer or an SME) only uses a standard WWW browser.

5. RELATED WORK

The 'Virtual Project Database' approach described in this paper aims at the integration of heterogeneous and distributed data sources to the illusion of a single uniform repository. Its core, the Virtual Database Kernel, is related to the Federated Database System (FDBS) concept (Sheth and Larson, 1990). A FDBS integrates multiple and heterogeneous, primarily relational and/or object-oriented, databases for uniform access via a global schema while the data are still managed in local autonomous database systems with local schemas. This is in contrast to Data Warehousing (LGI, 2000), where data from various data sources are collected and copied to a dedicated database (perhaps restructured for better suitability to analysis purposes), the 'data warehouse'.

Examples for recent FDBS research are Härder et al. (1997), Schönhoff et al. (1997), Frankhauser et al. (1998), Carey et al. (1995), or Radeke et al. (1997). The IDEE project (Integration of Data in Engineering Environments, Schönhoff et al., 1997) is of special interest since they also focus on engineering data (although in another domain) and also integrate tool-specific repositories and file systems besides standard DBMS into their federation concept. Their current focus is on global version management for versioned engineering objects (see e.g. Schönhoff and Strässler, 1998 and 1999).

Our Virtual Project Database approach is distinguished from traditional FDBS through clustering of the distributed data sources by Project Servers (one per site / organization) and intra-cluster coupling via an XML/HTTP protocol. This provides integration beyond company boundaries with an easy and flexible setup. A single Project Server, if used only locally, would be a 'simple' FDBS.

A approach contrary to federating heterogeneous data sources is managing all data by one single tool (which of course is not best-suited for all different types of data and does not provide support for legacy systems). That is the concept of the Engineering Data Management / Product Data Management (EDM/PDM) tools used in numerous engineering environments (Vital et al., 1998). In order to support distributed development processes with different PDM tools used by the participating companies, Abramovici et al. (1998) describe an agent based approach to federating heterogeneous and distributed PDM systems.

Cross-organizational workflows in general are investigated by Klingemann et al. (1999), targeting at general-purpose workflow management systems (WfMS). Salo and Käkölä (1999) highlight a single aspect of distributed development processes (requirements management) and describes a groupware (Lotus Notes) based requirements management system. Our approach applies a generic messaging paradigm and a out-of-line document link concept to generally support workflows related to engineering artifacts, leveraging the Virtual Project Database mechanisms. This combination implements the overall concept of a basic Virtual Project House infrastructure for engineering projects under the aspect of information sharing and exchange.

The term 'virtual project house' is also used by Schmidt et al. (1999). This paper discusses the logical composition of virtual project houses as sets of 'V-Labs' (virtual laboratories) as an organizational concept for Concurrent Engineering. Each V-Lab corresponds to a task and a special team of developers with a specialized tool

environment. The paper also describes a framework which is some kind of CSCW (Computer-Supported Cooperative Work, Teufel et al., 1995) environment for supporting asynchronous and synchronous communication and coordination in distributed developer teams including awareness of participants. The framework has to be filled up by integration of engineering and general groupware tools. The interface of the framework serves as an entrance point to the tool environment and graphically represents the virtual project house with 2-dimensional rooms for the V-Labs and icons for active team members which can move between the V-Labs.

6. CONCLUSION

This paper presented a 'Virtual Project House' infrastructure aiming at supporting the collaboration of engineers in distributed development processes (e.g. of electronic systems), especially in manufacturer-supplier-relationships. Due to the central role of design artifacts (product data), our approach concentrates on data sharing and data-centric workflows across site and organizational boundaries.

Founded on an integrated object model with well-defined global object identities and relationships a 'Virtual Project Database' integrates local and remote project data sources to a uniform information space. A general messaging concept, customized to workflows derived from a set of project work scenarios, is added on, facilitating structured information exchange among distributed developer teams.

The Virtual Project House system is composed of a distributed group of 'Project Servers' and a 'Project Browser' client application. The integration interface between different organizations (server to server) is based on Internet standards (XML for data serialization, HTTP for communication). The software itself is designed as platform-independent and realized in Java technology. There is no limitation to the number of participating partners. In addition to a fully bilateral architecture, an external partner can be integrated in a satellite-like manner, providing Extranet access to an in-house system via a WWW interface.

A main design principle is the generic approach and the open architecture. Extensions are possible by incorporating additional data management tools, supporting additional data types, viewers and editors, and by implementing additional scenarios / workflows.

7. REFERENCES

1. Abramovici, M., Gerhard, D., Langenberg, L., "Unterstützung verteilter Entwicklungsprozesse durch EDM/PDM" ("Support for distributed development processes through EDM/PDM", in German), Informationsverarbeitung in der Konstruktion '98, VDI Berichte Nr. 1435, 69-86, 1998.
2. Bray, T., Paoli, J., Sperberg-McQueen, C.M. (Eds.), "Extensible Markup Language (XML) 1.0", W3C Recommendation, <http://www.w3.org/TR/1998/REC-xml-19980210>, 1998.
3. Carey, M.J., et al., "Towards Heterogeneous Multimedia Information Systems: The Garlic Approach", Proceedings RIDE-DOM '95, Fifth International Workshop on Research Issues in Data Engineering - Distributed Object Management, Taipei, Taiwan, March 6-7, 1995.
4. Fallside, D.C. (Ed.), "XML Schema Part 0: Primer", W3C Working Draft (work in progress), <http://www.w3.org/TR/xmlschema-0/>, 2000.
5. Frankhauser, P., Gardarin, G., Lopez, M., "Experiences in Federated Databases: From IRO-DB to MIRO-Web", Proc. 24th VLDB Conference, New York, USA, 1998.

6. Frick, G., Sax, E., Müller-Glaser, K.D., "Knowledge Sharing across Organizational Boundaries with Application to Distributed Engineering Processes". In *Challenges of Information Technology Management in the 21st Century*, Mehdi Khosrowpour, ed. (Proceedings of the 2000 IRMA International Conference). Hershey, PA, USA: Idea Group Publishing, 2000.
7. Frick, G., Müller-Glaser, K.D., "Distributed Engineering – Virtual Design Houses", Technical Note FZI-ESM-VVE-TN-2, Forschungszentrum Informatik (FZI), Karlsruhe, Germany, 1999.
8. Gosling, J., Joy, B., Steele, G., "The Java™ Language Specification", Reading, MA: Addison-Wesley, 1996.
9. Härder, T., Sauter, G., Thomas, J., "Design and Architecture of the FDBS Prototype INFINITY", Proc. Internat. CAiSE '97 Workshop on Engineering Federated Database Systems (EFDBS '97), pp. 57-68, 1997.
10. Hyperwave, <http://www.hyperwave.com>, 2000.
11. International Business Machines (IBM), <http://www.ibm.com/software/ad/features/xmi.html>, 2000.
12. Klingemann, J., Wäsch, J., Aberer, K., "Deriving Service Models in Cross-Organizational Workflows", Proc. Research Issues in Data Engineering – Information Technology for Virtual Enterprises (RIDE-VE '99), Sydney, Australia, March 1999.
13. LGI Systems Inc., "A Definition of Data Warehousing", The Data Warehousing Information Center, <http://www.dwinfocenter.org/defined.html>, 2000.
14. Mertens, P., Griesse, J., Ehrenberg, D., "Virtuelle Unternehmen und Informationsverarbeitung" (Virtual Enterprises and Information Management, in German), Berlin – Heidelberg : Springer, 1998.
15. Perenthaler, E., Weichel, B., Hirth, T., Rauleder, P., "MSR-Standards in der Praxis" ("MSR standards in practical use", in German), VDI-Tagung, Baden Baden, Germany, <http://www.msrwg.de/badenbaden/MSR-in-der-praxis.htm>, 1998.
16. Radeke, E., Kruschinski, R., Nolte, D., "OpenDM/Web: WWW Access to Heterogenous Database Systems", Proc. 2nd Internat. Symposium on Global Engineering Networking, Antwerp, Belgium, pp. 141-155, April 1997.
17. Salo, A., Käkölä, T., "Groupware Architecture for Requirements Processes in New Product Development", Proceedings of the 32nd Hawaii Int. Conference on System Sciences, 1999.
18. Schmidt, K., Wagner, J., Bumiller, J., "A Cooperative Framework to Support Concurrent Engineering", 2nd Int. Conference on Concurrent Engineering in Construction (CEC 99), 1999.
19. Schönhoff, M., Strässler, M., Dittrich, K.R., "Data Integration in Engineering Environments", Proc. Internat. CAiSE '97 Workshop on Engineering Federated Database Systems (EFDBS '97), 1997.
20. Schönhoff, M., Strässler, M., "Issues of Global Version Management in Federated Database Systems for Engineering Environments", 10. Workshop Grundlagen von Datenbanken, Konstanz, Germany, June 1998.
21. Schönhoff, M., Strässler, M., "Global Version Management for a Federated Turbine Design Environment", Proc. NGITS '99, LNCS vol. 1649, Springer-Verlag, pp. 203-220, 1999.
22. Sheth, A., Larson, J., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys, vol. 22, no. 3, Sept. 1990.
23. Sun Microsystems, EJB, <http://java.sun.com/products/ejb/>, 2000.
24. Sun Microsystems, JSP, <http://java.sun.com/products/jsp/>, 2000.
25. Teufel, S., Sauter, C., Mühlherr, T., Bauknecht, K., "Computerunterstützung für die Gruppenarbeit" ("Computer-supported cooperative work", in German), Bonn: Addison-Wesley, 1995.
26. Vital, E., Vilela, J.B., Baake, U., Haussmann, D., "Introduction of Virtual Product Development in practice", Informationsverarbeitung in der Konstruktion '98, VDI Berichte Nr. 1435, 41-56, 1998.
27. Yeong, W., Howes, T., Kille, S., "Lightweight Directory Access Protocol", IETF RFC 1777, 1995.

ⁱ Parts of this contribution are based on work funded by the government of Baden-Württemberg (Germany) and the Deutsche Telekom AG within the 'medi@' initiative.

ⁱⁱ As an alternative to the EJB architecture (without affecting the general approach), the application logic can be incorporated as a base library into the Integration Server, the Web Presentation Server and the (then 'fat') client application. It has been done so in our prototype implementation.