# Distributed STEP-Compliant Platform for Multimodal Collaboration in Architecture, Engineering, and Construction

V. Semenov, A. Bazhan, S. Morozov, Institute for System Programming of the Russian Academy of Sciences (ISP RAS), Moscow, Russia, (step@ispras.ru)

## Summary

This paper presents an innovative software platform OpenSTEP intended to build advanced distributed integrated systems and to conduct multidisciplinary collaborative projects in both academy and industry. The paper discusses an open system architecture, methodology, component library and CASE toolkit enabling the developers to build a wide range of interoperable applications and systems compliant with STEP and, particularly, with IFC becoming the increasingly important standard for information integration in architecture, engineering and construction.

The component-based organization of the OpenSTEP platform assumes flexible capabilities to build systems with the client-server architecture, to configure and to deploy them in heterogeneous infrastructures of enterprises as well as to support effective collaboration of individuals and groups involved in joint projects. The achieved unification of the software interfaces and components provides a relatively easy migration path from existing single-site applications to advanced distributed collaborative environments and has potential making for investment succession.

The work is supported by RFBR (grant 04-01-00527) and Russian Foundation for Science Support (http://www.science-support.ru).

## 1 Introduction

The STEP is a family of international Standards for the Exchange of Product Model Data developed by the ISO Technical Committee 184 “Industrial automation systems and integration” since the middle of 80’s (ISO 1994). The objective of the STEP project is to provide standardized mechanisms for specification of product models as well as for computer-interpretable representation and exchange of product data in the ways neutral to potential software implementation platforms and applications.

Recently the STEP has been widely acknowledged as an underlying methodology approach and practical solutions valuable within Product Data Management (PDM) and Continious Acquisition and Life Circle Support (CALS) technologies. Following the STEP, interoperability between various CAD/CAM/CAE, ERP/MRP, CRM, and PDM applications can be achieved. To significant degree the STEP follows the model-driven methodology approach successfully exploited in different software engineering technologies, including the innovative Model-Driven Architecture (MDA) initiative by the OMG group (OMG 2002).

The International Alliance for Interoperability (IAI) utilizes the STEP infrastructure to develop and to promote own information schemas, namely, Industry Foundation Classes (IFC) enveloping a lot of aspects and disciplines the architecture, engineering, construction, and facility management industry (AEC/FM) is composed of. Since 1995 when IAI was born, several IFC versions have been released covering such domains as architecure, electrical appliance, heating, ventilation and air conditiong, construction management, facility management, cost estimation (IAI 1999).

Nevertheless, there are still remaining serious obstacles that prevent introduction of the STEP/IFC-based integration solutions into the industry practice. Mainly, they are connected with restricted capabilities of existing legacy applications to support these emerging standards, to form distributed integrated environments and to support effective collaboration between project participators.

Indeed, the existing CAD/CAM/CAE applications support only IFC file exchange that is dominating but not satisfactory way to achieve interoperability between the applications. Integration of applications through project model servers (Adachi 2002, Anumba 2003, Froese et al. 2000, Hannus et al. 2001, Zarli, Richaud 1999) may turn out superfluously expensive as these solutions are usually developed and delivered as monolithic systems exploiting particular software technologies and not allowing configurable deployment in heterogeneous infrastructures of enterprises and simple adaption to concurrent engineering practices preferable for particular user groups. These circumstances don’t make for investment succession and prevent factual dissemination of STEP/IFC standards.

Another adjacent problem is that the STEP is not comprehensive itself. Strongly focusing on interoperability, it omits many aspects vitally important for building, deploying and exploiting STEP-compliant systems. In particular, the following aspects are beyond the standard scope:

- The kinds of persistent stores for project model data and possible internal organization schemas to represent such information in general-purpose databases (relational, object-oriented, XML-based) and document repositories. Since the most known persistent realizations have application workloads at which they exhibit superiour performance, the general mechanisms have to be defined to connect up and to employ the alternative datastores especially when it may be crucial for effective collaboration;
- The methods for adaptation of the standard data access interfaces STEP SDAI to application programming interfaces (API) of particular systems to be integrated and involved in the distributed collaborative environments. The alternative SDAI binding versions and EXPRESS-X language regulated by the STEP seem to be not effective practical solutions for mapping the software interfaces;
- The methodology how to extend the early approved STEP information standards into permanently arising distributed system technologies as well as methods for porting of the legacy STEP-compliant applications into innovative middleware platforms like CORBA/IDL, J2EE/Java, XML/SOAP, .NET/C#, WebServices/WSDL. The specification of the SDAI at the EXPRESS data modeling language contains potential risk of missmatching the interface bindings and bridging mechanisms for different implementation platforms;
- The reference architectures for the distributed integrated systems incorporating the STEP-imposed capabilities for data exchange and sharing, as well as cooperative business logics for project model servers and alternative transaction models meaningful for today’s concurrent engineering practices and industry needs. Note that the STEP defines the only transaction model that is very restrictive for those cases in which the information schema encompasses the whole project information like in IFC;
- The general context of the STEP usage and its relationships with elements of PDM and CALS technologies like change, version, configuration, workflow, and document management.

The OpenSTEP project conducted at ISP RAS aims at improving general implementation infrastructure of the STEP (eliminating or compensating the disadvantages pointed above) as a result of usage of model-driven software engineering approach and at developing a software platform suitable for building a wide range of distributed integrated systems in different academy and industry branches, particularly, in AEC/FM domain. The OpenSTEP has a status of open software project implying that separate components can be developed, tested and delivered by the third-parties interested in validating and promoting the project into industry.

In the section 2 we first present an overview of the OpenSTEP platform principles and architecture in general and possible system configurations. Then, in the section 3 we concentrate on the platform organization, its kernel, and component libraries highlighting key engineering decisions that were made in the platform design and implementation. This section is also devoted to classification of transaction models allowed by the platform. The models are classified by transaction isolation levels and concurrency degree they provide. The section 4 concerns the accompanied CASE toolkit to automate development of software applications. In the Conclusions we shortly summarize advantages of the presented OpenSTEP platform and give references on the available solutions.

## 2 OpenSTEP overview

### 2.1 General principles

The presented OpenSTEP platform is a software component library and a CASE toolkit that, being applied together, have to enable the developers to build STEP-compliant applications and to integrate them within distributed collaborative environments. As such declaration of the goals is general enough to allow a lot of feasible solutions, the realization of the software project strongly follows the defined underlying principles and concrete technical requirements:

- Consistency of the software engineering solutions in conformity to a wide range of applications, including separate software components, stand-alone applications and distributed integrated systems;
- Comprehesiveness and effectiveness of the offered solutions to develop applications in highly automated and unified way using the same methodology approach, reference architecture, reused component library and CASE toolkit;
- Compliance of the developed applications with the STEP standards, including standard data exchange formats (Part21, Part28) and SDAI data access interfaces (Part22);
- Support of multimodal collaboration sessions implying that the particular users and groups can communicate and interact with each other under specific transaction models corresponding to particular concurrent engineering practices;
- Virtualization of data sources meaning that the alternative persistent stores of project model data and access providing services can be exploited by already developed applications without any rewriting and adapting their codes;
- Virtualization of communication environments meaning that the developed client and server applications can communicate with each other using alternative distributed middleware platforms and communication technologies;
- Allowing the alternative software interfaces to integrate particular third-party applications in addition to the standard STEP SDAI bindings;
- Mobility, interoperability and scalability of the developed applications as well as flexible capabilities to configure, to adjust, and to deploy them in heterogeneous environments, including both Windows and Unix/Linux platforms;

- Finally, the OpenSTEP platform must have an open system architecture enabling the developers to functionally evolve in future both the offered solutions and the built applications in the directions coincident to the emerging model-driven software engineering technologies.

## 2.2 Reference architecture for distributed STEP-compliant systems

The main intent of applications based on the presented OpenSTEP platform consists in providing the services to access project model data and utilizing them during collaborative multi-modal sessions. Systems with the client-server architecture are considered as main targets of the software platform. At the same time, building the other distributed systems, in particular, peer-to-peer architecture systems is also allowed reusing the same platform solutions.

The project model data defined by such multidisciplinary schemas as IFC may be very large and complicated. Often, they are composed of the huge number of highly granulated and closely related data units that are requested very intensively by participating applications. These peculiarities may be crusial issue affecting on feasibility of collaboration sessions. Therefore, there are implementation challenges and performance trade-offs relating to the STEP-based integration solutions. Nevertheless, the key decisions made about the OpenSTEP architecture, its design and implementation seem to be well balanced to satisfy both general-purpose integration needs and high-performance requirements assuming real time interactions.

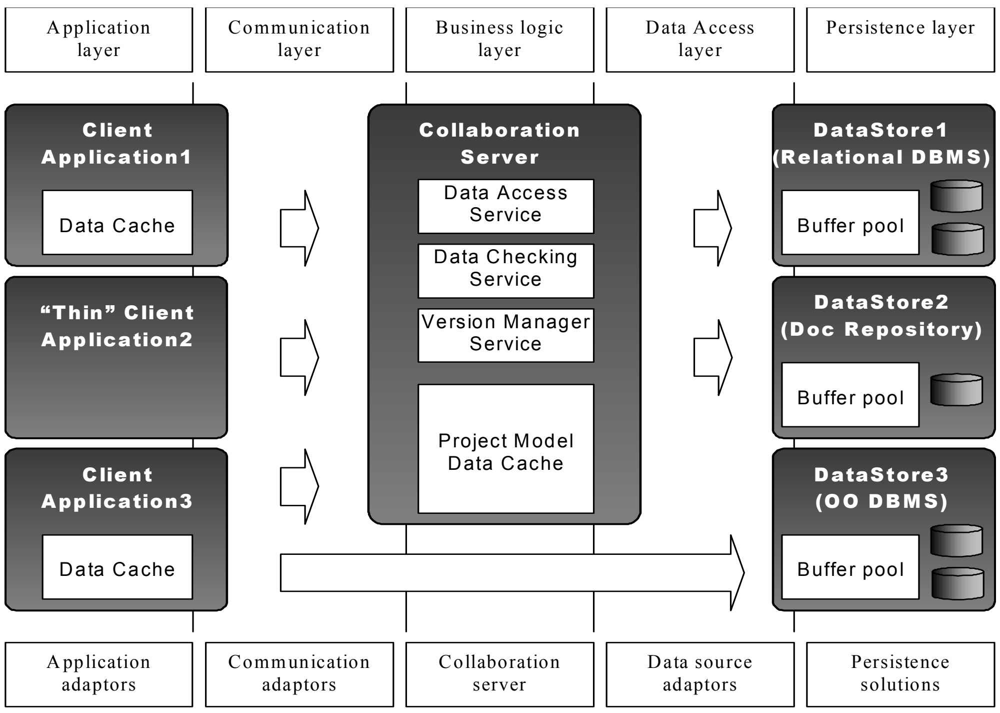

Figure 1. The reference architecture for STEP-compliant distributed integrated systems

The taken decisions are addressed to building the distributed integrated systems based on the multilayer client-server architecture presented at the figure 1. By functions the reference architecture encompasses the logical layers of persistent storing and archieving project model data (persistence layer), accessing to the datastores via software interfaces (data access layer), supporting multimodal collaboration sessions (business logic layer), delivering the provided collaboration services to participating applications (communication layer), and adapting these services to application-specific data and operations (application layer).

The persistence layer is represented by datastores allowing keeping the project model data in long-time stable storages. They may be both general-purpose relational, object-oriented or hybrid databases, document repositories, and specialized STEP-oriented solutions. It is desirable, but not necessary, the stores be capable to resolve data requests within project models.

The data access level assumes the facilities that provide access to the project model data via software interfaces. The software interfaces are suggested to be structured to suit to the logical organization of the model repository and to the underlying information schemas. An important feature realized at this level is a virtualization of datastores meaning that the applications can access project information via the same interfaces independently on internal logical and physical organization of the exploited storages.

The business logic layer is implemented by a collaboration server. Its intent consists in supporting the collaboration sessions and providing the appropriate services to participating applications. The server maintains own local caches for model data and is capable to execute all the data requests from clients. Sometimes it executes the request, in turn sending requests for the particular data items to the persistent stores. The server is an actual owner of project model data and ultimately responsible for preserving the integrity of data and enforcing the multimodal transaction semantics. It also controls the stores on which the permanent version of the project data resides. In addition to the underlying data access functionality, the collaboration server can potentially offer the other attendant services to check consistency of the project data upon constraints imposed by formally defined information schemas, to manage data changes and model versions, to produce visual representations of the project data, etc.

The purpose of the communication layer is to make the collaboration services offered by the remote server available to local applications by bringing them through some communication environment. The communication environment can be based on popular distributed middleware platforms like RPC, CORBA, J2EE, .NET, SOAP, WebServices. The realized virtualization principle implies that the participating client applications can interact with the collaboration server via the same software interfaces using alternative communication solutions.

And, finally, the application layer is formed from the participating client applications. To be involved into the distributed collaborative environment, the applications have to be adjusted to the offered collaboration services. The applications may maintain own local caches of project model data allowing some request processing to be immediately performed at the clients. Thin clients can interact with the collaboration server via corresponding facilities provided by the middleware platform and internally incorporated by the collaboration environment.

The considered reference architecture for the STEP-compliant distributed integrated systems employs query-shipping and data-shipping approaches that are of principal value for database management systems. Although the data-shipping approach provides both performance and scalability improvements, there is still much debate about the relative advantages of these models with respect to current technology trends and particular applications. The presented reference architecture permits combining and employing both models within particularly configured systems.

The interactions of *Client Application 1* with *Collaboration Server*, *Collaboration Server* with *DataStore 2* represented by a document repository as well as *Client Application 3* with *DataStore 3* that is suggested to be an object-oriented database are acomplished according to the undelying data-shipping model. In these interactions the client components determine which data items are needed to satisfy a given application request and obtain those items from servers if they cannot be found locally. Data caching enables clients to retain received copies of data items, to accelerate navigation through them and to offload much of the server functions to the client applications.

The interactions of "thin" *Client Application 2* with *Collaboration Server* as well as *Collaboration Server* with *DataStore 1* represented by a relational database correspond to the typical query-shipping model. The client components form requests for servers and receive those data items immediately needed for their subsequent processing.

### 2.3 System configurations

A lot of particular system configurations are allowed by this reference architecture that envelop various practical cases the applications can exchange and share project model data. According to the architecture the applications can interact with each other through the collaboration servers as well as through the shared central datastores. Due to virtualization of data sources the applications request heterogeneous datastores and various collaboration services via the same software interfaces, which provides additional flexibility in system building, configuring and deploying. At the same time, it is admitted that some applications can combine both client and server functionalities, thus, allowing more advanced system configurations.

The covered configurations support a lot of use cases the project model data can be exchanged and shared by the applications. These include file-based data exchange, application-to-application data exchange, server-based applications and multiple data repositories. For briefness, we omit detailed discussion, addressing to comprehensive descriptions and motivations conducted for AEC/FM industry needs in (Froese et al. 2000).

## 3 OpenSTEP platform

### 3.1 General organization

The OpenSTEP platform provides software engineering solutions to build distributed integrated systems based on the reference architecture considered above. Therefore, the platform organization presented follows its multilayer structure to significant degree (see the figure 2). The platform is organized into the following component groups: an unified kernel providing general-purpose components reused by various applications, patterns specific for collaborating client and server applications as well as three sorts of adaptor components. These are datastore adaptors handling the persistence of project model data, communication adaptors bringing the data and exposing the services to client applications as well as application adaptors adjusting particular applications to the commonly shared project model data. All the adaptors are implemented as plug-in components that can be added, removed or substituted within already developed and deployed applications, thus, satisfying to the declared virtualization principles.

Reusing the OpenSTEP platform solutions, a wide range of software applications can be developed and integrated. These are collaboration servers providing functionally rich services to share and to manage project model data, client applications employing such services via SDAI interfaces as well as standalone applications just supporting the export/import capabilities for files in the STEP-defined formats. The platform allows building and exploiting the "thin" client applications bypassing intermediate SDAI components and directly addressing to the offered collaboration services.

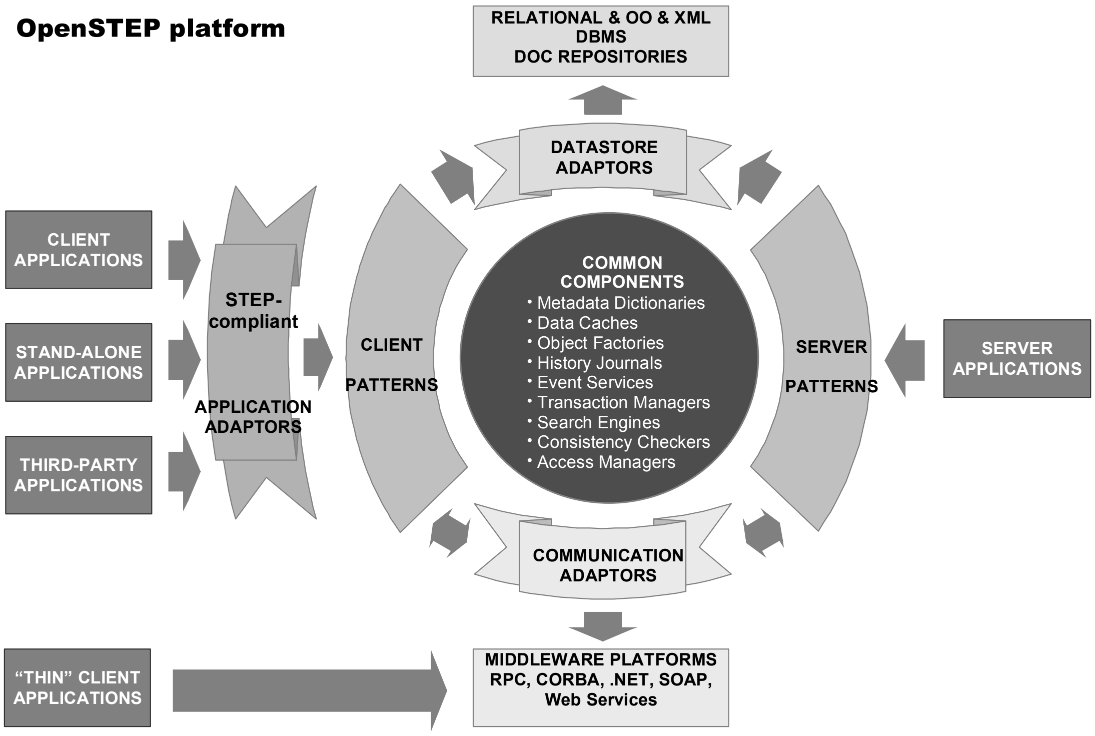

Figure 2. General organization of the OpenSTEP platform

## 3.2 Kernel

The kernel includes general-purpose components such as metadata dictionaries, object factories, data caches, transaction managers, search engines, data consistency checkers, and access managers. The components are generic in the sense that their implementation is independent on particular information schemas and specific collaboration modes. Genericity of the components with respect to various information schemas is achieved by means of reflection mechanisms and intensive usage of metadata. Genericity of the components in conformity to different collaboration modes is caused by a wide family of transaction models supported by the kernel.

To simplify development of the applications, the kernel provides patterns for typical client and server applications. The server pattern is a ready-to-use application template providing data management services and supporting multimodal collaboration sessions. The client pattern is a software library which should be linked into the user's application to involve it in the distributed integrated system. The server application is necessary multi-threaded so that it can handle requests from multiple clients and it uses separate processes to immediately resolve them or to redirect them to the assigned persistent datastores. Each application being linked with the client library requests to its local client component which executes the request, in turn sending requests for transaction support and for specific data items to the server. Each application can access to multiple servers from a single client process. The clients and server applications are usually executed on separate machines, but it is possible to run any number of clients and/or servers as separate processes on the same machine.

The OpenSTEP patterns provide with full support for both single- and multi-version concurrency control and recovery mechanisms constituting underlying ACID (atomicity, concurrency, isolation, durability) principles for distributed computing environments (Ramamritham, Chrysanthis 1997). They are realized by the components responsible for data caching and transaction managing that implies locking, handling history journals, recovery/rollback, event subscription and notification. In order to ensure that the replicated data caching in client applications does not result in violation of transaction semantics, the components handle the work on communicating and coordinating the client's caches with the caches located on the collaboration server.

For such purposes the components employ a transactional cache consistency maintenance algorithm. Plenty of such algorithms have been proposed in the literature (Franklin et al. 1997). The avoidance-based callback locking algorithm has been implemented in view of its uniformly high efficiency with respect to a wide range of workloads. Another important reason is the capability to update cached data in the clients as soon as the modifications have been committed by the other transactions which is essential for real time interactions.

## 3.3 Transaction models

In general, transactions in STEP-compliant applications tend to access many highly granulated data units interrelated with each other, may involve lengthy computations, and may be interactive pausing for input from users.

STEP SDAI defines the only transaction model. It permits the transactions to have shared access to whole model in reads mode or exclusive access to the whole model in writes mode. Such transactions degradate the potential concurrency due to increased data contention, in that way failing to meet the performance requirements. Such transactions are not satisfactory for engineering activities assuming simultaneous manipulation with the whole IFC project model data by different AEC/FM stakeholder applications.

Therefore, there is a need to exploit concurrency at lower granularity levels than the model level, to support more flexible transaction models with varied isolation levels and to take alternative correctness criteria not necessary reducing to traditional serializability. Running the concurrent transactions at different granularity and isolation levels allows applications to trade off concurrency and throughput for correctness. Lower levels increase transaction concurrency at the risk of allowing the transactions to observe fuzzy or incorrect data. If the latter is the case, validation routines have to be used to rollback unfortunate transactions or to correct the acquired data after the doubtful transactions have committed. In conformity to STEP-related data such corrections have to be applied to those data items that disturb transaction semantics or violate consistency constraints imposed by EXPRESS schemata.

In an effort to enhance concurrency in STEP-compliant applications, the OpenSTEP platform supports several alternative transaction models both pessimistic and optimistic ones at different granularity levels. The supported transaction models enable the users to hold sessions in the ways suitable for today's engineering practices. By choosing the appropriate model the users can achieve reasonable compromise between permanent maintaining the project data in the consistent state and effective concurrent engineering.

The figure 3 gives a set of transaction models allowed by the OpenSTEP platform. The models are arranged by transaction isolation levels and weakness criteria so the lower located models provide more concurrency but permit more anomalous phenomena in execution of transactions. These are well-known *Dirty Writes*, *Dirty Reads*, *Non-Repeatable Reads*, *Lost Update*, *Phantoms* phenomena as well as *Skew Reads*, *Skew Writes* phenomena relating to constraint violation anomalies and peculiar for optimistic multiversion concurrency control (Berenson et al. 1995).

The *Standard* model is a transaction model defined by the STEP SDAI. It is based on pessimistic locking policy (marked as "P" opposite to optimistic models marked as "O") at the SDAI model level (denoted as "M" opposite to "I" corresponding to the SDAI entity instance level). The model is placed at the top of the table as most restrictive and least concurrent one. At the bottom of the table optimistic multiversion transactions *Versioned* are represented. They suggest the separate transactions are executed independently starting from the fixed model versions. The commitment of transactions results in forming the new versions to be merged further by the methods resolving arisen ambiguities and compensating possible inconsistencies.

| Transaction models & Isolation levels | Type & Granularity Level | Dirty Write | Dirty Read | Fuzzy Read | Phantom | Lost Update | Read Skew | Write Skew |
|---|---|---|---|---|---|---|---|---|
| Standard (S) | (P & M) | X | X | X | X | X | X | X |
| Serializable | (P & I) | X | X | X | X | X | X | X |
| Repeatable read (S) | (P & I) | X | X | X | | X | X | X |
| Snapshot | O & I | X | X | X | | X | X | |
| Consistent read (S) | O & I | X | X | | | | | X |
| Read committed (S) | P & I | X | X | | | | | |
| Read uncommitted (S) | P & I | X | | | | | | |
| Versioned | O & M | | | | | | | |

Figure 3. OpenSTEP transaction models categorized by isolation levels and concurrency degrees

Finally, the middle part of the table is represented by the transaction models that proceed at the SDAI entity instance granularity level. The pessimistic models, namely, *Serializable*, *Repeatable Read*, *Read Committed* and *Read Uncommitted* correspond to the isolation levels introduced by the ANSI/ISO SQL-92 specification (ANSI 1992). These models are closely related with the behaviour of lock managers and differ in the ways how the acquired locks on separate entity instances and predicate locks on instance populations are released: just after completion of the operation or at the end of the transaction. The *Serializable* isolation level correponds to the classical serializability definition and avoids any phenomena from the considered collection. The other isolations admit some phenomena which are marked in the figure.

The *Snapshot* and *Read Consistency* models are multiversion optimistic models often used in the popular database systems. The first model follows so-called "first-committer-wins" policy that requires the system to remember all the updates belonging to any transaction that commits after the start-timestamp of each active transaction. It aborts the transaction if its updates conflict with the remembered updates by others. In the cases where short update transactions conflict minimally and long-running transactions are likely to be read-only, this model should give good results. The *Read Consistency* isolation follows "first-writer-wins" policy giving each request the most recent data value at the time the statement began and aborting the transactions conflicting with the remembered updates by other transactions.

The transaction models supported by the current version of the OpenSTEP platform are specially marked by the symbol "S" in the table. The other models are planned to be supported in the next versions.

## 3.4 Communication adaptors

The OpenSTEP platform provides CORBA-based implementation of the described communication environment called a communication adaptor. As it was pointed above the communication adaptor plays role of an integration bus bringing the services and data of the collaboration server to the local applications and emulating the work with remote STEP SDAI objects residing at the server as with own local ones, i.e. with their proxies in the clients.

The client part of the adaptor is implemented as CORBA stubs and the server part — as CORBA skeleton realizations (OMG 2004). This approach is extended to all the high-level semantics concepts like repository, schema instance, model. For objects corresponding to low-level semantics concepts like entity instance and instance attribute, direct copying the data is applied using CORBA valuetypes and custom marshalling. Such implementation feature is motivated by the need to unload the general-purpose ORB facilities from the specific work on handling the huge number of data units.

Seemingly, this approach is applicable to the implementations of communication adaptors using the other distributed middleware technologies. The OpenSTEP project has a target to explore alternative technologies like .NET and Web Services in order to validate the underlying methodology approach and software solutions. If there is a need to connect client and server applications running on heterogeneous middleware platforms, corresponding bridges between the platforms have to be additionally established within the communication adaptors.

## 3.5 Datastore adaptors

Datastore adaptors are the components responsible for persistence of the project model data in external storages. The current implementation of the OpenSTEP platform provides an advanced library of adaptors satisfying to wider requirements to represent, to store, and to access the project model data defined by EXPRESS schemas. These are the adaptors to document repositories based on STEP Part 21 and Part 28 exchange file formats as well as the adaptors to some relational databases Oracle, PostgreSQL, MySQL.

The repository adaptors realize access to project model data contained in the Part 21 and Part 28 documents as well as provide the incorporating applications with simple export/import capabilities. The adaptors are not capable to execute requests, therefore such processing has to be performed by applications. Although Part 28 format allows exploiting XML database technologies, this capability requires more careful investigations and performance estimates for STEP benchmarks.

The database adaptors support different database design strategies, namely, late binding, early binding and BLOB strategies. All three strategies are static in the sense that the assumed database schemas don't depend on the state of really stored data.

The late binding approach results in a relational table system suitable for any project model data driven by EXPRESS information schemas. The relational system includes tables semantically associated with STEP SDAI concepts as well as auxiliary tables intended to execute requests by retrieving both the stored data and attended metadata. The advantage of this strategy consists in the limited number of the relational tables and the stored procedures applicable to arbitrary information schemas.

The early binding strategy consists in mapping each particular information schema into a database schema and representing the project model data by the relational tables specific for the source schema. The advantage of this strategy is more efficient execution of typical requests for the entity instances by persistent identifiers, subtyping relations, logical predicates, association relations. The serious drawback is the huge number of tables making inconvenient the database administration for such complicated EXPRESS schemas as IFC releases and potentially degradating the total performance.

The BLOB strategy follows the late binding approach resulting in the database schema independent on particular project models with the exeption of the entity instance representations. All the entity-specific attributes are packed and represented by binary or textual records similar to Part 21 strings or Part 28 tags. In such strategy the requests for entity instances by identifiers and subtyping relations are executed efficiently on condition that packing and unpacking of the records are accomplished by the client applications. The other requests have to be entirely resolved by the applications.

Thus, the implemented library supports a variety of datastores and database strategies which allow choosing those solutions that are most suitable for specific collaboration purposes and performance reqirements. Following the undelying virtualization principle, the datastore adaptors can be employed by target applications in any combinations.

### 3.6 Application adaptors

The application adaptors are intended to involve applications into distributed collaborative environments. A standard solution is the support of SDAI interfaces via which the separate applications can access to the project model data. The implemented library of application adaptors realizes C++ early, late and mixed binding versions assumed by the STEP standard. The features of these versions are mainly related to signatures of the methods supplied for entity classes. In the early binding version these methods correspond to entity definition in the source information schema, in the late binding version they are completely unified with respect to different schemas, and the mixed binding version combines the methods admitted by both the previous approaches.

Since the integrated applications typically manipulate with own data, there is a need to map the application-specific data into commonly shared project model representation and vice versa. For these purposes the OpenSTEP offers implementation patterns that allow handling bi-directional maps and maintaining both application-specific and common data representations consistently. The patterns are regarded as alternative implementations of the application adaptors in addition to the STEP SDAI-compliant components.

## 4 OpenSTEP toolkit

Besides the component libraries, the OpenSTEP platform provides a CASE toolkit intended to improve software engineering activities connected with development of data management facilities in the ways following the early binding approach. As information schemas like IFC are very complicated and may contain hundreds of data types and constraints imposed upon them, the CASE tools enable the developers to significantly simplify the implementation of corresponding components. The OpenSTEP toolkit is a family of EXPRESS translators and accompanied utilities that unify and automate typical software engineering processes.

Concerning the discussed issues, the toolkit provides translators for generating ready-to-use STEP SDAI-compliant application adaptors at C++ language as well as datastore adaptors to relational databases with all the necessary concomitant schemas, packages of queries and stored procedures. Development of EXPRESS translators into Java and C# languages and corresponding application adaptors is also within the project tasks.

## 5 Conclusions

Being assembled in different combinations, the OpenSTEP solutions presented above allow building a lot of STEP-compliant applications. These are standalone applications with export/import facilities for STEP files, applications integrated through central databases as well as advanced distributed systems with the multilayer client-server architecture.

Due to the achieved virtualization of data sources the applications can be flexibly configured to encompass a variety of the use cases meaningful for AEC/FM industry needs. Allowing alternative communication environments and running under both UNIX and Windows platforms, the OpenSTEP applications can be easily deployed within and across heterogeneous information infrastructures of enterprises. Supporting multimodal collaboration, the applications can be adjusted to specific user requirements and concurrent engineering practices.

The conducted unification of the software interfaces and components provides a relatively easy migration path from existing single-site applications to advanced distributed collaborative environments and has potential making for investment succession. The platform has been successfully used to implement the OpenSTEP collaboration server and the client applications supporting the whole family of IFC schemas. Now these solutions are free available from the project web site http://www.ispras.ru/~step for validating and promoting into industry practice.

## 6 References

Adachi, Y. (2002). Overview of IFC model server framework. In European Conference of Product and Process Modelling (ECPPM'02), eWork and eBusiness in AEC, eds. Turk, Z. and Scherer, R.

ANSI, (1992). ANSI X3.135-1992, American National Standard for Information Systems — Database Language — SQL, November, 1992.

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., O'Neil, P. (1995). A Critique of ANSI SQL Isolation Levels. In SIGMOD Record Conference, San Jose, CA USA, 1995, Vol. 24, No. 2, pp. 1-10.

Franklin, M., Carey, M., Livny, M. (1997). Transactional Client-Server Cache Consistency: Alternatives and Performance. In ACM Transactions on Database Systems, Vol. 22, No. 3, September 1997.

Froese, T., Yu, K., Liston, K., Fischer, M. (2000). System architectures for AEC interoperability. In Proc. International Conference on Construction Information Technology, Reykjavik, Iceland, 2000, Vol.1, pp. 362-373.

IAI, (1999). IFC Technical Guide, International Alliance for Interoperability, <http://www.iai.org.uk/documentation/IFC_2x_Technical_Guide.pdf>

ISO, (1994). ISO 10303: 1994, Industrial automation systems and integration — Product data representation and exchange.

Kazi, A., Hannus, M., Laitinen, J., Nummelin, O. (2001). Distributed engineering in construction: findings from the IMS GLOBEMEN project. In ITcon, Vol. 6 (2001), pp. 129-148.

OMG, (2002). OMG Model Driven Architecture: How systems will be built. <http://www.omg.org/mda>

OMG, (2004). CORBA/IIOP Specification, v3.0.3 http://www.omg.org/cgi-bin/apps/doc?formal/04-03-12.pdf

Owolabi, A., Anumba, C., El-Hamalawi, A. (2003). Architecture for implementing IFC-based online construction product libraries. In ITcon, Vol. 8 (2003), pp. 201-218.

Ramamritham, K. and Chrysanthis, P. (1997). Executive Briefing: Advances in Concurrency Control and Transaction Processing. IEEE Computer Society Press, 1997.

Zarli, A., Richaud, O. (1999). Requirements and technology integration for IT based business-oriented frameworks in building and construction. In ITcon, Vol. 4 (1999), pp. 53-74.