

IMPLEMENTATION OF STANDARDIZED SHAREABLE PRODUCT DATABASES

Vinícius Medina Kern, Rogério Almeida Barra, and Ricardo Miranda Barcia

Article submitted to the Second International Congress of Industrial Engineering, Piracicaba, SP, Brazil, October 7-10, 1996.

Contact:
Vinícius Medina Kern
vmkern@cme.nist.gov

IMPLEMENTATION OF STANDARDIZED SHAREABLE PRODUCT DATABASES

Vinícius Medina Kern
PPGEP-UFSC and NIST
vmkern@cme.nist.gov

Rogério Almeida Barra
Fundação Centro Tecnológico para Informática
barra@ia.cti.br

Ricardo Miranda Barcia
Programa de Pós-Graduação em Engenharia de Produção
UFSC-CTC-EPS - Caixa Postal 476 - 88040-900 - Florianópolis-SC, Brasil

Abstract

The Standard for the Exchange of Product model data (STEP, or ISO 10303) is a family of standards aimed to represent a product's information throughout its entire life cycle. STEP comprises several classes of documents organized in a database-like architecture. This paper focuses on STEP Implementation Methods and their application on the implementation of shareable product databases. File exchange, the simplest STEP Implementation Method, is described. SDAI, a language-independent application programming interface for access and manipulation of STEP data, is discussed, together with an outline of its application on the integration of network interoperable product databases.

Keywords

STEP, SDAI, Engineering Databases, EXPRESS, Product Data Exchange (PDE), Interoperability, CORBA

1. Introduction

The purpose of the Standard for the Exchange of Product model data (STEP, or ISO 10303) is “to prescribe a neutral mechanism capable of completely representing product data throughout the life cycle of a product” [1].

STEP is a family of standards, comprising several classes of documents organized in a database-like architecture. This paper focuses on STEP Implementation Methods and their application on the implementation of shareable product databases, a core technology for the integration of engineering applications.

The architecture adopted for STEP development is presented, along with the role played by the Implementation Methods. Four implementation levels devised for STEP are presented. File exchange, the simplest STEP Implementation Method, is described. SDAI, a language-independent application programming interface for access and manipulation of STEP data, is

STEP document series	Document part numbers	Architectural layer
Introductory	single digits	
Description Methods	10 series	
Implementation Methods	20 series	Physical layer
Conformance Testing Methodology and Framework	30 series	
Integrated Resources (IRs)	40 and 100 series	Logical layer
Application Protocols (APs)	200 series	Application layer
Application Interpreted Constructs	500 series	
Abstract Test Suites	300 series	

Table 1 - STEP documents series and architectural organization

discussed, and its application on the integration of network interoperable product databases is outlined.

2. An architectural view of STEP implementation

STEP is a collection of standards organized in classes or series of documents according to a three-layer architecture, i.e., *application*, *logical*, and *physical* layer. Table 1 displays the three layers and other complementary series.

To produce a STEP implementation, an implementation method, defined in the physical layer, is combined with an Application Protocol (AP), an application-specific information model specified in the application layer. A description of information models is given in [2]. This paper focuses on STEP implementation, with an emphasis on shareable databases. Both information model and implementation method specifications are written in EXPRESS, STEP Part 11 [3].

The implementation methods define the mapping of an application schema onto a specific computer technology. Four progressive implementation levels were devised:

- Level 1: File exchange -- passive text file transfer
- Level 2: Working form exchange -- software assisted, active file transfer
- Level 3: Database exchange -- shared database access
- Level 4: Knowledgebase exchange -- integrated knowledge-based systems

Levels 1 and 2 are well developed, and level 3 has research and commercial implementations. Level 4 implementations are subject of basic research. The next sections discuss STEP implementation methods and their importance in the implementation of shareable engineering databases.

3. File exchange and working form

STEP Part 21 [4] defines an application-independent way of encoding product data for file exchange. Figure 1 presents an example of Part 21 usage, with EXPRESS schema definitions and an exchange file with data structures compliant to the schemas.

The STEP implementation level 1 defines a static text file transfer. A STEP pre-processor in the sending system converts the product data model to an ASCII physical file format. This file is transferred to a receiver system, which converts standard data into its internal representation using a post-processor. Both pre- and post-processor must comply with Part 21.

(a)	(b)
SCHEMA example_geometry; (* edited for brevity *)	ISO-10303-21;
-- short name:	HEADER;
ENTITY cartesian_point -- cpt	/*
SUBTYPE OF (point);	edited for brevity
x_coordinate : length_measure;	*/
y_coordinate : length_measure;	FILE_SCHEMA(('EXAMPLE_GEOMETRY',
z_coordinate : OPTIONAL length_measure;	'EXAMPLE_TOPOLOGY'));
END_ENTITY;	ENDSEC;
END_SCHEMA;	DATA;
SCHEMA example_topology;	/*
REFERENCE FROM example_geometry;	THE FOLLOWING 13 ENTITIES REPRESENT A
ENTITY vertex -- vx	TRIANGULAR EDGE LOOP
SUBTYPE OF (topology);	*/
vertex_point : OPTIONAL point;	#1=CPT(0.0,0.0,0.0); /* CARTESIAN POINT ENTITY */
END_ENTITY;	#2=CPT(0.0,1.0,0.0);
ENTITY edge -- ed	#3=CPT(1.0,0.0,0.0);
SUBTYPE OF (topology);	#11=VX(#1); /* VERTEX ENTITY */
edge_start : vertex;	#12=VX(#2);
edge_end : vertex;	#13=VX(#3);
END_ENTITY;	#16=ED(#11,12); /* EDGE ENTITY */
ENTITY edge_logical_structure -- ed_strc	#17=ED(#11,13);
edge_element : edge;	#18=ED(#13,12);
flag : BOOLEAN;	#21=ED_STRC(#17,.F.); /* EDGE LOGICAL
END_ENTITY;	STRUCTURE ENTITY */
ENTITY edge_loop -- ed_loop	#22=ED_STRC(#18,.F.);
SUBTYPE OF (loop);	#23=ED_STRC(#16,.T.);
loop_edges : LIST [1:?] OF edge_or_logical;	#24=ED_LOOP(#21,#22,#23)); /* EDGE LOOP ENT. */
END_ENTITY;	*/
END_SCHEMA;	NOTE: OTHER SYNTACTIC REPRESENTATIONS FOR
	A TRIANGULAR EDGE LOOP WERE POSSIBLE.
	EACH REPRESENTATION MAY CARRY DIFFERENT
	SEMANTICS, WHICH MAY REPRESENT
	CHARACTERISTICS OF PARTICULAR APPLICATION
	SYSTEMS.
	*/
	ENDSEC;
	END-ISO-10303-21;

Figure 1 - EXPRESS schema definitions (a) and an exchange structure example (b) [4]

According to Fowler [5], implementation level 2 is an extension to Level 1, where the physical file is converted to a “working form.” Since most file-based implementations use this approach, the distinction between levels 1 and 2 have disappeared.

Part 21 files allow for data exchange and archiving of data models comprising entire STEP schemas. Implementation levels 1 and 2, using STEP Part 21, are the simplest implementation specifications. However, since it may not always be possible to rely on the availability of more advanced implementations, Part 21 implementations are very useful. A number of public tools are available that support the building of applications using implementation levels 1 and 2 [6] [7].

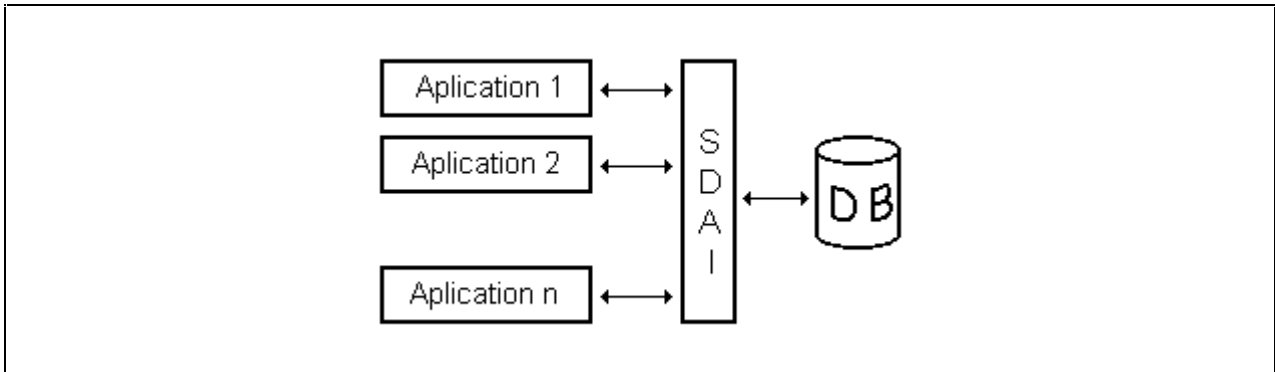


Figure 2 - Single-database STEP data sharing

Implementation levels 3 and 4 provide access to data at a finer level of granularity, and represent a different approach: they address not only data *exchange*, but also dynamic data *sharing*.

4. STEP database implementations with SDAI

The Standard Data Access Interface (SDAI) is an application programming interface (API) that allows for CAX ('Computer-Aided anything') systems to access, manipulate, and share STEP data. SDAI provides low-level operators to create, retrieve, and delete objects, and also to insert, update, and delete attributes.

The interface defined in the SDAI is based on a conceptual model. Different database technologies can be used for data storage without the need to alter the interfaces used between the applications and the database [5]. With SDAI, database developers do not have to worry about how each application handles data, and application programmers can use SDAI functions without concern about the details of the underlying database technology.

Figure 2 presents a schematic view of STEP data sharing using a single database: applications access STEP data in a common database through SDAI. Data must comply with an information model defined in an application protocol (AP). Each application must provide a pre- and postprocessor to translate between its own data format and the AP.

SDAI is said to *isolate* applications from the data storage technology. Goh et al. [8] comment on how SDAI can be used to share data between different applications and databases:

- Each database's application programming interface provides the means by which the data independence is perceived by the applications. However, these interfaces differ from database to database. Such differences impede data sharing and exchange.
- SDAI addresses this issue by means of a consistent set of mechanisms, regardless of the underlying data storage technology.
- Any application whose data is modeled in EXPRESS can utilize SDAI, thereby making the actual database transparent.

SDAI specifications are written in EXPRESS. Since EXPRESS is not intended to be implemented, SDAI has several bindings to programming languages, such as C [9], C++ [10], and Fortran. Implementations vary in binding style, i.e., the way in which the database software is structured [11]:

<u>Input</u>	
Transaction:	sdai_transaction; The transaction allowing access to entity instances which is to be terminated.
<u>Possible error indicators</u>	
SS_NOPN	Session not open.
FN_NAVL	Function not available.
TR_NEXS	Transaction does not exist.
TR_EAB	Transaction ended abnormally.
TR_NAVL	Transaction not available.
SY_ERR	Underlying system error.
<u>Effect on the SDAI environment</u>	
Transaction shall be deleted.	
The active_transaction attribute of the currently open sdai_session shall be unset.	
schema_instance.change_date for any modified or created schema_instance shall be set to the current date.	
sdai_model.change_date for any modified or created sdai_model shall be set to the current date.	

Figure 3 - Specification of the SDAI operation *End transaction access and commit* [12]

- Early binding, or code generation: the application is generated for a specific information model written in EXPRESS.
- Late binding, or data dictionary: the implementation is a general purpose software that is independent of information model, allowing access to data defined by *any* EXPRESS information model.

Figure 3 presents an example of SDAI functionality. The operation *End transaction access and commit* ends the sequence of operations started by *Start transaction*, and commits (makes persistent) all changes to the entity instances made since the most recent *Start transaction*.

Besides the standard mappings to programming languages, there is work addressing the mapping of EXPRESS into the relational standard query language, SQL [13]. However, the relational model does not have the richness of EXPRESS. It is expected that object-oriented languages and databases will yield better STEP implementations.

SDAI does not specify details about concurrent access control and transaction management. These issues are to be handled by each database management system. It also does not support connections to remote repositories. The SDAI binding to CORBA IDL [14] addresses this problem. The integration of STEP and CORBA allows for network interoperable access to STEP data, described below.

5. Network interoperable databases

Interoperability is defined as “the effective interconnection of two or more different computer systems, databases, or networks in order to support distributed computing and/or data exchange” [15]. Applications in a distributed computer system interoperate with each other, but execute in separate address spaces, possibly on different hardware/software platforms.

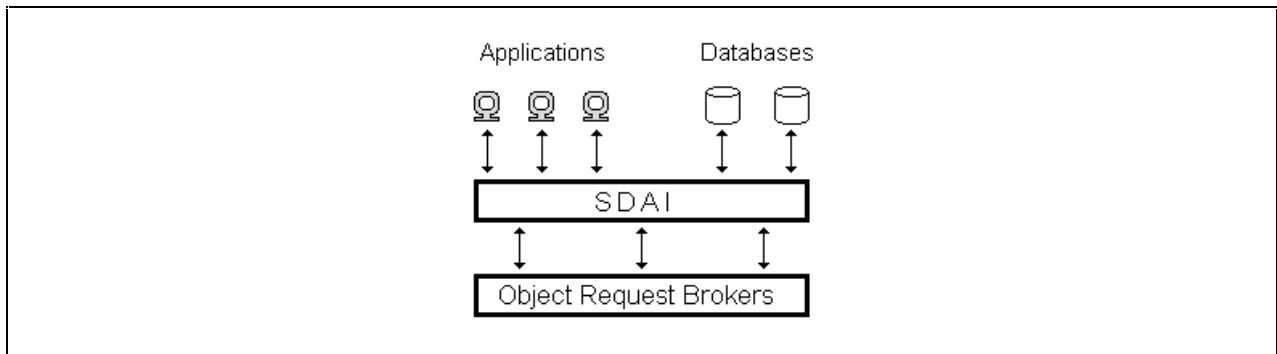


Figure 4 - Network interoperable access to STEP databases using CORBA

The approach adopted for STEP network interoperability comes from the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [16]. CORBA specifies an architecture whose core is the Object Request Broker (ORB). ORBs enable objects to make and receive requests and responses in a distributed environment. Objects made available through an ORB publish their interfaces using CORBA's Interface Definition Language (IDL).

In order to integrate STEP and CORBA, ISO has released a draft standard of the SDAI/IDL binding [14]. This will allow for STEP data models to be shared among applications and databases in a network interoperable environment.

Figure 4 illustrates the network interoperable access to STEP data. End-user applications and STEP databases may be in different locations and/or networks, connected to different SDAI implementations and ORBs. A data request from an application is passed on to an ORB. The ORB communicates with other ORBs in order to locate the database. A database application receives the request and, using database operators translated from the SDAI operators, performs the access and delivers data back to the application.

The adequacy of specific database technologies for complex applications like CAx systems has been the object of intense debate. Stone and Hentchel observed, regarding this 'database war':

“How do you decide which type of database is best when even the experts can't agree?”
[17]

Even if an optimal point solution can be found, it is unlikely that it would be good enough for every application in a distributed computing environment. The implementation of network interoperable product databases will rather depend upon a combination of different systems and technologies. The integration of STEP and CORBA allows for different database approaches to be integrated, provided that they can map their data structures in accordance to the conceptual schemas defined in STEP APs.

6. Conclusion

In this paper, we described STEP's implementation methods and their use on the implementation of shareable product databases. The integration of SDAI with CORBA, using a standard binding from SDAI to CORBA IDL, will allow for the integration of network interoperable product databases.

The implementation of the standardized, shareable, network interoperable STEP databases is vital for the integration of engineering applications, and further for the integration of industrial enterprises, in both vertical and horizontal aspects: concurrent engineering and virtual enterprises.

References

- [1] Esther Edwards-Iwe, "A Client/Server Implementation of the Design Process Using PDES/STEP 'Level 3' Data Sharing Architecture," In: *Engineering Data Management: Key to Success in a Global Market*, Proceedings of the 1993 ASME International Computers in Engineering Conference and Exposition, San Diego CA, 1993, pp. 15-23.
- [2] Vinícius M. Kern, Jan Helge Bøhn and Ricardo M. Barcia, "The Building of Information Models in STEP," article submitted to the *II International Congress of Industrial Engineering*, Santa Bárbara d'Oeste, SP, Brazil, October 7-10, 1996.
- [3] ISO TC184/SC4/WG5 Document N65 (P2), *Product Data Representation and Exchange Part 11, EXPRESS Language Reference Manual*, ISO International Standard, November 1, 1994.
- [4] ISO DIS 10303-21, *Product Data Representation and Exchange Part 21, Implementation Methods, Clear Text Encoding of the Exchange Structure*, Draft International Standard, May 28, 1993.
- [5] Julian Fowler, *STEP for Data Management, Exchange and Sharing*, Technology Appraisals Ltd., Twickenham UK, 1995, 214 pp.
- [6] Don Libes, *The NIST STEP Part 21 Exchange File Toolkit: An Update*, National Institute of Standards and Technology, NISTIR 5187, 1993. Available by anonymous ftp at <ftp.cme.nist.gov/pub/step/nptdocs/p21tk-update.ps>
- [7] Stephen Nowland Clark, *The NIST Working Form for STEP*, National Institute of Standards and Technology, NISTIR 4351, November 19, 1990. Available by anonymous ftp at <ftp.cme.nist.gov/pub/step/nptdocs/stepwf.ps>
- [8] A. Goh, S.C. Hui, B. Song and F.Y. Wang, "A Study of SDAI Implementation on Object-Oriented Databases," *Computer Standards & Interfaces*, 16 (1), 1994, pp. 33-43.
- [9] ISO TC184/SC4/WG7 Document N394, *Product Data Representation and Exchange Part 24, Standard Data Access Interface - C Language Late Binding*, ISO Committee Draft, July 28, 1995.
- [10] ISO TC184/SC4/WG7 Document N403, *Product Data Representation and Exchange Part 23, C++ Programming Language Binding to the Standard Data Access Interface*, ISO Committee Draft, December 25, 1995.
- [11] Martin Hardwick and David Loffredo, "Using EXPRESS to Implement Concurrent Engineering Databases," In: *Proceedings of the Computers in Engineering Conference and the Engineering Database Symposium* (Eds: Ahmed Busnaina and Ravi Rangan), ASME, Boston, MA, September 17-20, 1995, pp. 1069-1083.
- [12] ISO TC184/SC4/WG7 Document N382, *Product Data Representation and Exchange Part 22, Standard Data Access Interface*, ISO Committee Draft, May 31, 1995.
- [13] Katherine C. Morris, *Translating EXPRESS to SQL: A User's Guide*, National Institute of Standards and Technology, NISTIR 4341, 1990.
- [14] ISO TC184/SC4/WG7 Document N396, *Product Data Representation and Exchange Part 26, Interface Definition Language Binding to the Standard Data Access Interface*, ISO Working Draft, December 7, 1995.

- [15] Office of Science and Technology Policy / Federal Coordinating Council for Science, Engineering, and Technology, *High Performance Computing & Communications: Toward a National Information Infrastructure*, Report by the Committee on Physical, Mathematical, and Engineering Sciences, Washington, D.C., 1994, 176 pp.
- [16] Object Management Group, *The Common Object Request Broker Architecture: Architecture and Specification*, Revision 2.0, July 1995.
- [17] Christopher M. Stone and David Hentchel, "Database wars revisited," *Byte*, October 1990, pp. 233-242.