Chapter 7

ONTOLOGIES- AND STANDARDS-BASED APPROACHES TO INTEROPERABILITY FOR CONCURRENT ENGINEERING

L. C. Pouchard, A.F. Cutting-Decelle

7.1. INTRODUCTION

As the use of information technology and computer-driven systems in manufacturing and construction design has matured, the necessity for software applications to work together, exchange data, processes and information has become crucial to the conduct of business and operations in organisations. This capability is referred to as interoperability (Pouchard 2000), (Pouchard 2002), (Ray 2003). To be competitive and maintain good economic performance, organisations need to employ increasingly effective and efficient data and computer systems. Such systems should result in the seamless integration of application data and exchange of processes between applications. Organisations should also be able to conserve and retrieve on demand the knowledge contained in their business and operational processes, regardless of the applications used to produce and handle these processes.

With the increasing need for enterprise integration, developers face more complex problems related to inter-operability. Independent contractors and suppliers who collaborate on demand within virtual supply chains must share product-related data. Vendor applications that are not designed to inter-operate must now share processes. When enterprises collaborate, a common frame of reference or at least a common terminology is necessary for human-to-human, human-to-machine, and machine-to-machine communication. Similarly, within a core enterprise where distributed collaboration between remote sites and production units take place, a common understanding of business- and manufacturing-related terms is indispensable. However, this common understanding of terms is often at best implicit in the business transactions and software applications and may not even be always present. Misunderstandings between humans conducting business-related tasks in teams, and ad-hoc translations of software applications contribute to the rising costs of interoperability in manufacturing.

Standard-based approaches and ontologies offer a direction addressing the challenges of interoperability brought about by semantic obstacles, i.e. the obstacles related to the definitions of business terms and software classes. An ontology is a taxonomy of concepts and their definitions supported by a logical theory (such as first-order predicate calculus). Ontologies have been defined as an explicit specification of a conceptualization (Gruber 1993). An ontology expresses, for a particular domain, the set of terms, entities, objects, classes and the relationships between them, and provides formal definitions and axioms that constrain the interpretation of these terms (Gomez-Perez 1998). An ontology permits a rich variety of structural and nonstructural relationships, such as generalization, inheritance, aggregation, and instantiation and can supply a precise domain model for software applications (Huhns and Singh 1997). For instance, an ontology can provide the object schema of object-oriented systems and class definitions for conventional software (Fikes, Farquhar 1999). Ontological definitions, written in a human readable form, can be translated into a variety of logical languages. They can also serve to automatically infer translation engines for software applications. By making explicit the implicit definitions and relations of classes, objects, and entities, ontology engineering contributes to knowledge sharing and re-use (Gomez-Perez 1998). Ontology engineering aims at making explicit the knowledge contained within software applications, and within enterprises and business procedures for a particular domain and includes a set of tasks related to developing ontologies for a particular domain.

Interoperability in manufacturing refers to the ability to share technical and business information seamlessly throughout an extended enterprise (supply chain) (Ray, Jones 2003). This information, previously shared in a variety of ways including paper and telephone conversations, must now be passed electronically and error-free with suppliers and customers around the world. A study, achieved by the NIST in 2002 (NIST, 2002) was aimed at identifying the economic impact of the use of standards in industry, particularly the ISO 10303 STEP standard with the objective of conducting an economic impact assessment of STEP's use by transportation equipment industries, namely the automotive, aerospace, shipbuilding, and specialty tool and die industries. Both the full potential and current realized benefits are quantified. In addition, the study investigates the impact of NIST's administrative and technical contributions to STEP. The authors of the study estimate the economic value of the efficiency gains due to improved data exchange enabled by using STEP, and quantify NIST's contributions to those gains. Data collected from industry surveys and case studies are used to estimate the potential benefits of existing STEP capabilities. They estimate that STEP has the potential of save $928 million (in 2010) per year by reducing interoperability problems in the automotive, aerospace, and shipbuilding industries. Currently approximately 17 percent ($156 million) of the potential benefits of STEP quantified within the scope of this study are being realized. A previous study commissioned by NIST (NIST, 1999) in 1999, had reported that the U.S. automotive sector alone expended one billion dollars per year to resolve interoperability problems. The study also reported that as much as 50% of this expenditure is attributed to dealing with data file exchange issues.

7.2. INTEROPERABILITY IN CONSTRUCTION: WHAT DO WE MEAN?

7.2.1 Information Systems in Construction: Specificity and Main Features

Information systems become increasingly important in industrial companies for acquiring, structuring, and exchanging complex technical data that they have to handle during the production process. The intrinsic complexity of the information becomes yet more complex with the relational structuring of the data. This structure is necessary in order to select among the set of possible solutions the most competitive ones in answer to given specifications. This is particularly true for construction SMEs, since they are often exposed to situations for which they have neither the necessary skills nor the tools enabling a continuous updating of the technical information needed by the projects they work on and the software tools they use (Cutting-Decelle, Dubois, in Bestougeff et al., eds, 2002).

Fundamentally, the construction industry is characterised by:

- an increasing complexity with an acceleration of the relations among the partners, particularly in a CE context, alongside a dramatic reduction of the lead-time between the call for tender and the operation of the building (Anumba et al. 1999) ;
- an increasing diversity of the information and data handled, mainly due to the development of new representation structures (use of standard messages such as EDIFACT messages, use of product data -de facto or de jure- standards: STEP, P-LIB and IFCs, as we will see in this chapter;
- the development of new software tools capable of dealing with the increasing volume and diversity of information, although, most of the time, without any interoperability between them ;
- a great heterogeneity of the information handled, since a normal construction project requires several documents simultaneously. These include drawings, calculation, technical notes, bills of materials and other kinds of technical analysis, as well as documents (legal or not) containing information related to the different building components.

The evaluation of the degree of elaboration of an information system starts with the possibility to identify and to interface, when possible, existing document repositories or product databases, regardless of their structuring and location.

7.2.2. The Concept of Interoperability

Interoperability is 'the ability of software and hardware on multiple machines from multiple vendors to communicate.' (FOLDOC)

“Interoperability” is considered as “the ability of a system or a product to work with other systems or products without special effort on the part of the customer. Interoperability becomes a quality of increasing importance for information technology products as the concept that "The network is the computer" becomes a reality”. (Whatis)

For P. Miller (Interoperability, 2000) : “to be interoperable, one should actively be engaged in the ongoing process of ensuring that the systems, procedures and culture of an organisation are managed in such a way as to maximise opportunities for exchange and re-use of information, whether internally or externally.”

Based upon this definition, it should be clear that there is far more to ensuring interoperability than using compatible software and hardware. Rather, assurance of effective interoperability will require often radical changes to the ways in which organisations work and, especially, in their attitudes to information.

Different approaches to the challenges of interoperability exist. One is likely to find them in combination in real-world problems:

- A standard-based approach : the most straightforward aspect of maintaining interoperability. Consideration of technical issues includes ensuring an involvement in the continued development of communication, transport, storage and representation standards. Work is required both to ensure that individual standards move forward to the benefit of the community, and to facilitate where possible their convergence, such that systems may effectively make use of more than one standards-based approach.

- A software engineering approach: In this approach, software developers and quite often users who need the data outputs of an application as input to another write some syntactic parsers that allow the language and/or the data structures of the output to be mapped to the structures and language of the second application. This approach does not take into account the semantic conflicts and gaps described below. Furthermore, the mappings between two applications are ad hoc, that is left to the subjective understanding of concepts by developers. Finally, each time a new application or even a new version of an existing application occurs, the parsers need to be modified.

- A semantic interoperability approach : Semantic interoperability presents a host of issues, all of which become more pronounced as individual resources — each internally constructed in their own semantically consistent fashion — are made available through 'gateways' and 'portals'. Almost inevitably, these discrete resources use different terms to describe similar concepts (“slab”, “floor”, “level”, “surface”, for example), or even use identical terms to mean very different things, introducing ambiguïty and error into their use. This situation is troubling because the errors introduced are not necessarily explicit and may induce errors in analysis or design.
There are also other kinds of interoperability, among which we will mention : human interoperability, inter-community interoperability, legal interoperability. In this chapter, we will focus on technical interoperability among software tools used by the professionals of the construction sector.

7.2.3 The Need for Interoperability

Being seen to "be interoperable" is becoming increasingly important to a wide range of organisations, projects, even companies. In each case undeniably valuable information is being made available to a wide range of users, often for the first time. The drive towards interoperability will necessarily lead to changes in the way the organisations operate. One of the aims of this book is to show that concurrent engineering provides a valuable tool of the interoperability.

A truly interoperable organisation is able to maximise the value and reuse the potential of information under its control. It is also able to exchange this information effectively with other equally interoperable bodies, allowing new knowledge to be generated from the identification of relationships between previously unrelated sets of data.

The lack of interoperability is very costly to some industrial sectors. But changing internal systems and practices to make them interoperable is a non-trivial task. However the benefits for the organisation and those making use of information it publishes are potentially incalculable, as mentioned in the introduction to this chapter.

7.2.4 The Potential of Standards to Increase Interoperability

There are three principal approaches to compensate for the lack of interoperability:

The first is a point-to-point customized solution, which can be achieved by contracting the services of systems integrators. This approach is expensive since each pair of systems needs a dedicated solution.
A second approach, adopted in some large supply chains, requires all partners to conform to a particular solution. This approach does not solve the interoperability problem since the first or sub-tier suppliers are forced to purchase and maintain multiple, redundant systems. It can also be costly to the smaller organizations in a supply chain since they are rarely in a position to influence the choice of infrastructure, and may not have enough resources to comply.
The third approach involves neutral, open, published standards. By adopting open standards the combinatorial problems is reduced from n^2 to n, with bi-directional translators.

Published standards also offer some stability in the representation they propose of the information models, an essential property for long-term data archiving. This chapter highlights some of the standards developed within the ISO TC184 “Industrial Automation Systems and Integration” Committee, particularly those relevant to the construction sector (ISO).

But the problem is far from solved. Interoperability standards are used in layers, from the cables and connectors, through networking standards, to the application or content standards such as those mentioned here, that is STEP, P-LIB and PSL (Process Specification Language). All of these layers must function correctly for interoperability to be achieved. The greatest challenges remain at the top of this stack of standards, in order to make them inter-operable. Due to the capability of the PSL language to be extended (through its ontology) for accommodating concepts in other standards, this language can be considered as a powerful tool of this interoperability, enabling, for a near future, the consideration of a “universal interfacing”.

We present in the following sections some of the main (de facto and de jure) standards that can be used in construction. Since this approach of the construction sector with an interoperability based on standards is rather new, we describe the most known in the domain of product data modelling (ISO 10303 STEP, ISO 13584 P-LIB and IAI/IFCs) but also a new standard used for the specification of process related information, the ISO 18629 PSL standard. This PSL language brings an important contribution to the problem of the semantic ambiguïty met in the information exchanges.

7.3. INTERNATIONAL STANDARDS DEVELOPED BY THE ISO TC 184 COMMITTEE

The ISO TC184 is one of the one two hundred committees managed by the ISO (International Standardisation Organisation, Geneva, CH) (ISO), its scope is : “Standardisation in the field of industrial automation and integration concerning discrete part manufacturing and encompassing the applications of multiple technologies, i.e. information systems, machines and equipments and telecommunications”. This means that the standards developed are applicable to manufacturing and process industries, applicable to all sizes of business, applicable to extending exchanges across the globe through e-business.
Are excluded from the scope the following domains : electrical and electronic equipment (dealt

with by the IEC/TC44) and programmable logical controllers for general applications (IEC/TC65). The scope of the committee means that the standards developed are : applicable to manufacturing and process industries, applicable to all sizes of business, applicable to extending exchanges across the globe through e-business.
The standards developed by the ISO TC184 and its different sub-committees cover various domains related to industrial automation and integration, among which: enterprise modelling, enterprise architecture, communications and processes, integration of industrial data for exchange, access and sharing, life cycle data for process plants, manufacturing management, mechanical interfaces and programming methods, part libraries, physical device control, Process Specification Language (PSL), product data, and robots for manufacturing environment (Cutting-Decelle et al, 2004-1).

7.3.1 ISO 10303 STEP

Each part of ISO 10303 contains the following introductory paragraph that summarizes the significant challenges undertaken in this standardization effort (Kemmerer, 1999):
"ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the lifecycle of a product, independent from any particular system. The nature of this description makes STEP suitable not only for neutral file exchange, but also as a basis for implementing, sharing product databases, and archiving (IS 10303-1, 1994)."

STEP was designed to be the successor of exchange standards such as IGES, SET, and VDA-FS with the notable difference that it was intended to do more than support exchange of product data. STEP is intended to support data sharing and data archiving. These distinguishing concepts are given below:

Product data exchange: the transfer of product data between a pair of applications. STEP defines the form of the product data that is to be transferred between a pair of applications. Each application holds its own copy of the product data in its own preferred form. The data conforming to STEP is transitory and defined only for the purposes of exchange.
Product data sharing: the access of and operation on a single copy of the same product data by more than one application, potentially simultaneously. STEP is designed to support the interfaces between the single copy of the product data and the applications that share it. The applications do not hold the data in their own preferred forms. The architectural elements of STEP may be used to support the realization of the shared product data itself. The product data of prime interest in this case is the integrated product data and not the portions that are used by the particular product data applications.
Product data archiving: the storage of product data, usually long term. STEP is suitable to support the interface to the archive. As in product data sharing, the architectural elements of STEP may be used to support the development of the archived product data itself. Archiving requires that the data conforming to STEP for exchange purposes is kept for use at some other time. This subsequent use may be through either product data exchange or product data sharing.

Early in the development of ISO 10303, SC4 (ISO TC184/SC4) recognized that the scope of the standard was extremely large. This fact resulted in a couple of fundamental assumptions that shaped the architecture of STEP. SC4 assumed it unlikely that any one organization would implement the entire ISO 10303, due to its large scope. Therefore, it made sense to separate the standard into parts, where an organization would implement only the subset of parts needed to satisfy the requirements of their operation. Another primary concept contributing to the architecture is that the content of the standard is to be completely driven by industrial requirements. This, in combination with the concept that the re-use of data specifications is the basis for standards, led to developing two distinct types of data specifications. The first type, reusable, context independent specifications, defines the building blocks of the standard. The second type, application-context-dependent specifications (application protocols) is developed to satisfy clearly defined industrial information requirements. This combination enables avoiding unnecessary duplication of data specifications between application protocols.

SC4 determined that computer-sensible standards specifications were necessary to facilitate reliability and efficiency. The expression of STEP data constructs through a formal data definition language is necessary (but not sufficient) for unambiguous definition of data.

7.3.1.1 Components of ISO 10303

The architecture of STEP is intended to support the development of standards for product data exchange and product data sharing. The requirements and concepts in the preceding section have contributed to the evolution of the architecture over the past decade. The architectural components of STEP are reflected in the decomposition of the standard into several series of parts. The STEP document composition was developed at the June 1989 meeting of ISO TC184/SC4/WG1 as a series of parts. Each part series contains one or more types of ISO 10303 parts. Figure 7.1 provides an overview of the structure of the STEP documentation.

[Insert Figure 7.1 here]

The following describes each of the structural components and functional aspects as an overview of the STEP architecture.

Description Methods : The first major architectural component is the description method series of STEP parts. Description methods are common mechanisms for specifying the data constructs of STEP. Description methods include the formal data specification language developed for STEP, known as EXPRESS (10303-11, 2004). Other description methods include a graphical form of EXPRESS, a form for instantiating EXPRESS models, and a mapping language for EXPRESS.

Implementation Methods : The second major architectural component of STEP is the implementation method series of 10303 parts. Implementation methods are standard implementation techniques for the information structures specified by the only STEP data specifications intended for implementation, application protocols. Each STEP implementation method defines the way in which the data constructs specified using STEP description methods are mapped to that implementation method. This series includes the physical file exchange structure (10303-21, 1994), the standard data access interface (10303-22, 1998), and its language bindings (10303-23, 2000), (10303-24, 2001), (10303-26, 1997). Implementation methods are standardized in the ISO 10303-20 series of parts.

Conformance Testing : The third major architectural component of STEP is in support of conformance testing. Conformance testing is covered by two series of 10303 parts: conformance testing methodology and framework, and abstract test suites. The conformance testing methodology and framework series of 10303 parts provide an explicit framework for conformance and other types of testing as an integral part of the standard. This methodology describes how testing of implementations of various STEP parts is accomplished.

Data Specifications : The final major component of the STEP architecture is the data specifications (see Figure 7.2). There are four part series of data specifications in the STEP documentation structure, though conceptually there are three primary types of data specifications: integrated resources, application protocols, and application interpreted constructs. All of the data specifications are documented using the description methods.

[Insert Figure 7.2 here]

** Integrated Resources : The integrated resources constitute a single, conceptual model for product data. The constructs within the integrated resources are the basic semantic elements used for the description of any product at any stage of the product lifecycle. Although the integrated resources are used as the basis for developing application protocols, they are not intended for direct implementation. They define reusable components intended to be combined and refined to meet a specific need. The integrated resources comprise two series of parts, the integrated generic resources and the integrated application resources. The two series have similar function and form: they are the application, context-independent standard data specifications that support the consistent development of application protocols across many application contexts.

** Application Protocols : Application protocols (APs) are the implementable data specifications of STEP. APs include an EXPRESS information model that satisfies the specific product data needs of a given application context. APs may be implemented using one or more of the implementation methods. They are the central component of the STEP architecture, and the STEP architecture is designed primarily to support and facilitate developing APs. Many of the components of an application protocol are intended to document the application domain in application specific terminology. Application protocols are standardized in the ISO 10303-200 series of parts.

** Application Interpreted Constructs : Application interpreted constructs (AICs) are data specifications that satisfy a specific product data need that arises in more than one application context. An application interpreted construct specifies the data structures and semantics that are used to exchange product data common to two or more application protocols. Application protocols with similar information requirements are compared semantically to determine functional equivalence that, if present, leads to specifying that functional equivalence within a standardized AIC. This AIC would then be used by both application protocols and available for future APs to use as well. STEP has a requirement for interoperability between processors that share common information requirements. A necessary condition for satisfying this requirement is a common data specification. Application interpreted constructs provide this capability. Application interpreted constructs are standardized in the ISO 10303-500 series of parts.

Since a few years, a new concept, called “common resources” has appeared within the STEP community. This concept is aimed at maximising the re-use of already existing elements, either directly within the data specifications, or by means of the development of “application modules” : see Figures 7.3, 7.4 and 7.5 for further information about the components of the standard.

[Insert Figures 7.3, 7.4, 17.5 here]

7.3.2 ISO 13584 P-LIB

7.3.2.1 Purpose

ISO 13584 (13584-1, 1999) specifies the structure of a library system which provides an unambiguous representation and exchange of computer interpretable parts library information. The data held in the library are a description that enables the library system to generate various representations of the parts held in the library. The structure is independent of any particular computer system and permits any kind of part representation. The structure will enable consistent implementations to be made across multiple applications and systems.
ISO 13584 does not specify the content of a supplier library. The content of a supplier library is the responsibility of the library data supplier. The library management system used in the implementation of the structure defined in ISO 13584, and any interface between this system and a user of the system is the responsibility of the library management system vendor and is not specified in ISO 13584.

7.3.2.2 Components of a Library System

The components which form a neutral library system may be split into a number of functional areas which are illustrated in Figure 7.6.

[Insert Figure 7.6 here]

User to computer system communication : The interface between the user and his computer system is not defined in ISO 13584. This would be application dependent and form part of the user interface supplied by a vendor as part of a computer system.

Interface to External Systems : The interface between a library system compliant with ISO 13584 and other software systems are :
a library Interrogation Interface : not defined in ISO 13584 but would be expected to provide facilities to select parts from the library and to define the orientation, position and representation category of the part selected ;
a representation transmission interface, enabling the library system to send parts representations to the user computer system ;
an input interface for library data, enabling the integration of supplier libraries within a library system.

7.3.2.3 Internal Structure of a Library System
A Library system consists of a Dictionary, Library Management System and Library Content as shown in Figure 7.7. The standard defines these modules by the requirements placed upon their functional behaviour. ISO 13584 does not standardise their implementation.

[Insert Figure 7.7 here]

- Dictionary : consisting in a set of entries associated with a human-readable and computer-sensible representation of the meaning associated with each entry. The dictionary may be accessed by the user and referenced from library data. The Dictionary provides a referencing mechanism between library data obtained from different suppliers and enables the user to obtain an understandable view of the parts held in the library. The dictionary structure is specified in ISO 13584-42 (13584-42, 1998). A supplier library may contain only dictionary entries. These entries provide computer-referable identifiers for the concepts involved in some application domain.

- Library Management System : software system that enables the end user of the library to use the content of an integrated library and to load data into that library. The Library Management System is not standardised within ISO 13584.

- Library content : Library data are structured into classes in accordance with the object oriented paradigm. Three kinds of classes are considered in ISO 13584. The contents of the three kinds of classes may be exchanged using the structure and exchange format specified in P-LIB.
General model classes enable library data suppliers to provide the definition of a collection of cognate parts considered as a part family. Functional model classes enable library data suppliers to provide various representations (e.g. geometric, schematics, procurement data etc.) for these collections of cognate parts. Functional view classes enable the specification of the kind of representation provided in the different functional model classes. Some functional view classes are standardised in the view exchange protocol series of ISO 13584. A library data supplier may also provide the definition of their own functional view class. These three kinds of classes are illustrated in Figure 7.8.

[Insert Figure 7.8 here]

When a library consists only of a dictionary, it only defines the concept associated with each class and with the properties of each class. When a library also contains a library content, this content defines the set of instances contained by each defined class.

When the user CAD system is compliant with an ISO 10303 STEP application protocol(s), the provisions contained in ISO 13584 ensure that it is possible from a library content to generate a functional view that is compliant with an ISO 10303 application protocol.

7.3.2.4 Fundamental Principles of the standard

ISO 13584 separates the representation of information held in a parts library from the implementation methods used in data exchange. The standard makes use of a formal data specification language, EXPRESS, to specify information about the structure of a library. ISO 13584 separates information about the structure of a parts library from the information about different representations of each part or family of parts in the library. ISO 13584 permits information about part representation to be specified by different standards, and includes mechanisms which enable references to such descriptions (Pierra et al. 1998), (Pierra et al. 2004).

7.3.2.5 Structure of the ISO 13584 series of parts

ISO 13584 is divided into series of parts, each with a unique function. Each series may have one or more parts. The series are listed below with their numbering scheme :

Conceptual Description	-	Parts 10 to 19
Logical resources	-	Parts 20 to 29
Implementation resources	-	Parts 30 to 39
Description Methodology	-	Parts 40 to 49
Conformance Testing	-	Parts 50 to 59
View Exchange Protocol	-	Parts 101 to 199
Standardised content	-	Parts 500 to 599

- Conceptual descriptions : they define the global conceptual framework and mechanisms developed to allow the portability of multi-supplier and multi-representation parts libraries, for exchanging and for updating. They present a problem domain analysis of the universe of discourse. They describe the concepts and choices made in the formulation of ISO 13584. The division of the whole task to be performed into a number of logical tasks that may be defined as a separate part of ISO 13584 is accomplished in the conceptual description series of parts.

- Logical resources : The information model of parts library is provided by a set of resources. Each resource is comprised of a set of data descriptions in EXPRESS, known as resource constructs. One set may be dependent on other sets for its definition. Some resources constructs from ISO 10303 may be used to define ISO 13584 resources constructs. All the ISO 13584 resource constructs are defined in one part of the logical resources series. These resources may be used, but not modified, in a view exchange protocol.

- Implementation resources : Each representation category may require a representation transmission interface to be implemented on a receiving CAD system to be able to interpret part models and to generate part views. The implementation resources specify the standardised representation transmission interfaces which may be referenced by a view exchange protocol. Each part of this series either specifies an interface, with the requirements for its implementation, or specifies the requirements for the implementation of one interface specified in other Standards.

- Description methodology : providing rules and guidelines for library data suppliers, who may be standardisation organisations, part suppliers or functional model suppliers. These rules are intended to ensure consistency of a user Library. They are mandatory for the standardisation committees, in charge of specifying standardised dictionary data. They provide optional guidelines for part suppliers or functional model suppliers.

- Conformance testing : providing test cases and a set of requirements that any implementation shall meet before being accepted as conforming to this Standard.

- View exchange protocol : specifying one set of requirements for the exchange of one representation category of parts. Several view exchange protocols may refer to the same representation category. A view exchange protocol may introduce different options that may be selected by an implementation. The options are termed conformance classes. In this case the requirements of the view exchange protocol are specified separately for each conformance class.

- Standardised content : It is intended to progressively define standardised dictionary entries which may be referenced by supplier libraries. This work will be done inside different standardisation committees following the methodology specified in the description methodology series of parts of ISO 13584. The parts of the standardised content specify the standardised dictionary entries corresponding to various application areas.

7.3.2.6 Use of library parts in product data

An ISO 13584 conforming exchange context provides for the exchange of library data intended to be stored in a user library. An ISO 10303 STEP conforming exchange context provides for the exchange of product data.

Three levels of interactions have been identified between these two levels of exchange.

- Level 1 : All information about a part generated in System A will be transferred to System B by means of ISO 10303 (see Figure 7.9).

[Insert Figure 7.9 here]

- Level 2 : Only that information is transferred from System A to System B, which is necessary to generate the same part from a Library 2 of the receiving System B at the required position and orientation. Library 1 and Library 2 both contain all the information about the part (see Figure 7.10).

[Insert Figure 7.10 here]

- Level 3: That information is transferred from System A to System B which is necessary to generate the same part information on the receiving System B without any assumption about the content of Library 2. This means that the transferred data also contains a subset of Library 1 (see Figure 7.11).

[Insert Figure 7.11 here]

The information models specified in ISO 13584 are intended to enable these three levels of interaction.

7.3.3 ISO 18629 : Process Specification Language : PSL

ISO 18629 is the newest in the family of standards aimed at facilitating interoperability for industrial data integration (of products and processes) in industrial applications in TC 184. Standardised within a joint committee, ISO TC 184 SC4/SC5, PSL provides a generic language for process specifications applicable to a broad range of specific process representations in manufacturing and other applications. PSL is an ontology for discrete processes written in the Knowledge Interchange Format (KIF) (Genesereth and Fikes, 1992) itself an ISO candidate in ISO/JTC1, (Common Logic, 2004). Each concept in the PSL ontology is specified with a set of definitions, relations, and axioms all formally expressed in KIF. Relations specify types of links between definitions or elements of definitions ; axioms constrain the use of these elements. In addition, the PSL ontology is based on set theory, first order logic, and situation calculus (Etchemendy 1992). Because of this reliance on theories, every element in the PSL language can be proven for consistency and completeness (Gruninger 2003). At the time of this writing, approximately half of the PSL definitions, relations and axioms have been proven to be consistent with the base theories.

PSL is an international standard for providing semantics to the computer-interpretable exchange of information related to manufacturing and other discrete processes. Taken together, all the parts contained in PSL provide a language for describing processes throughout the entire production within the same industrial company or across several industrial sectors or companies, independently from any particular representation model. The nature of this language makes it suitable for sharing process information during all the stages of production. The process representations used by engineering and business software applications are influenced by the specific needs and objectives of the applications. The use of these representation models varies from one application to another, and are often implicit in the implementation of a particular application. One of the manufacturing models on which the PSL ontology is built is provided by the information models of the ISO 15531 MANDATE standard (standardisation of manufacturing management information) (Cutting-Decelle et al., 2000-1), particularly for resource management.

A major purpose of PSL is to enable the interoperability of processes between software applications that utilise different process models and process representations. As a result of implementing process interoperability, economies of scale are made in the integration of manufacturing applications.

All parts in ISO 18629 are independent of any specific process representation or model used in a given application. Collectively, they provide a structural framework for interoperability. PSL describes what elements should constitute interoperable systems, but not how a specific application implements these elements. The purpose is not to enforce uniformity in process representations. As objectives and design of software applications vary the implementation of interoperability in a application must necessarily be influenced by the particular objectives and processes of each specific application.

7.3.3.1. Architecture and content of ISO 18629
PSL (ISO IS 18629-1, 2004) is organized in a series of parts using a numbering system consistent with that adopted for the other standards developed within ISO TC184/SC4. PSL contains Core theories (Parts 1x), External Mappings (Parts 2x), and definitional extensions (Parts 4x). This discussion focuses on Parts 1x and 4x ; these parts contain the bulk of ISO 18629, including formal theories and the extensions that model concepts found in applications. Parts 1x are the foundation of the ontology, Parts 4x contain the concepts useful for modeling applications and their implementation. Table 7.1 presents the organization of ISO 18629. Except noted otherwise, PSL version 2.2 is presented.

[Insert Table 7.1 here]

- Core theories (Parts 1x) :
Core Theories include the PSL-Core, the Outer-Core, Duration and Ordering theories, Resource theories, and Actor and Agent theories. The core theories are contained in the parts 1x and based on first-order logic. They model basic entities necessary for building the PSL extensions. The PSL Core and Core Theories pose primitive concepts (those with no definition), function symbols, individual constants,
and a set of axioms written in the language of PSL. Table 7.2 illustrates the primitives found in the PSL Core. These primitives and all the definitions in PSL are written in KIF for computer interoperability but the KIF writing is not shown here for the sake of readability. For KIF sentences expressing these relations and functions the reader is referred to the PSL Web site.

[Insert Table 7.2 here]

Core theories are required to formally prove that extensions are consistent with each other, and with the core theories. The core theories are at the root of the PSL ontology against which every item that claims to be PSL compliant must be tested for consistency. They are a unique feature of PSL as no other standard in SC4 lends itself to formal, logic-based proof. Figure 7.12 illustrates concepts in the PSL Outer Core and their dependencies.

[Insert Figure 7.12 here]

Figure 7.13 extends Figure 7.12 to focuses on Duration, Ordering, and Resource Requirements theories.

[Insert Figure 7.13 here]

Domain-specific Definitional Extensions (Parts 4x) : The extensions to the Core and Outer-core are used to represent the actual processes in an application. All terms in the extensions are given definitions using the set of primitive concepts axiomatized in the core theories. This ensures that definitions are consistent with PSL. A software application will typically use the concepts defined in the extensions, rather than the concepts in the Core and Outer core, which are necessary to define the extensions but have little expressivity.

In Figure 7.14, a definitional extension (Parts 4x) is represented as a slice of the pie. It specifies concepts and definitions for all kinds of (practical) concepts and are written using the Core, Outer core, and theories. Some definitional extensions also use concepts defined in other extensions. Figure 7.14 shows that a concept belonging to an extension (blue triangle) is specified using concepts of the Core, Outer Core, and another extension. But the Core and Outer Core alone are not sufficient to represent meaningfully an application's semantics for the purpose of interoperability.

[Insert Figure 7.14 here]

Table 7.3 gives examples of definitional extensions, and the core theories each extension relies upon. It is to be noted that the organization of the Extensions into Activity Extensions, Temporal Extensions, etc… is here for readability and ease of use of the standard. The organization itself does not affect the concepts in PSL: for instance a concept may be moved from one extension to another without affecting the PSL ontology or the concepts defined in the extension. In other words, to be a valid part of the PSL ontology extensions do not need to belong to one or another of the categories in the left column. However, each concept must conform to the Core Theories in the middle column.

[Insert Table 7.3 here]

7.3.3.2. Interoperability with PSL and conformance to the standard

The main purpose of PSL is to establish a computer language for exchanging processes between software applications such as CAD, and project design software. As a specification language, PSL can be considered as a specification tool of the information and knowledge related to manufacturing management, as modelled by the MANDATE standard (ISO IS 15531-1, 2002).

7.3.3.2.1 The challenges of interoperability

The obstacles to interoperability of data regarding syntax of two applications are common, and usually dealt with parsers. Obstacles due to semantic problems, i.e. problems about the "meaning" of a software object or entity are less visible. Lack of semantic reconciliation may introduce errors even if syntax mapping is correct. Without a standard like PSL, the semantic mapping may be performed in an ad hoc manner by a developer.

Figure 7.15 presents an example from the transportation industry, where a truck is represented as a vehicle, a mobile resource, or a truck. Delivery mechanisms not represented here may also include transportation for some applications. If there is no interoperability of processes, applications that use this terminology may be incompatible. This leads to re-inputting entries manually in the application chain. In the example in Figure 7.16, Material designates two different things: a Resource and a Work in Progress and a Resource. Resource a Material, a Machine-tool, and a

Stock.

[Insert Figures 7.15 and 7.16 here]

Syntactic interoperability does not resolve these conflicts, and decisions as to which concept in Application A matches a concept in Application B is left to the developer of parsers. The benefit of PSL is to formally encode each application's concept or vocabulary in a rigorous representation language. When two applications sharing data are expressed in PSL, the conflicts and semantic gaps are highlighted and a resolution is proposed. In essence, expressing the concepts of an application with PSL produces a detailed analysis of processes, and on this basis two applications can be reconciled.

7.3.3.2.2 Interoperability and conformance
From the point of view of ISO 18629 two applications can inter-operate if they are conformant with the same set of ISO 18629 extensions. Software applications that claim conformance to PSL will:
specify processes from their application into the KIF language. This is the set of terms used by the application that refer either to processes in the application or relations among these processes.
provide translation definitions between their processes represented in KIF and PSL definitions.
implement syntactic translators between their applications and PSL process descriptions.
Another requirements not discussed in this chapter is that there exists a grammar using the same representation as PSL grammar for the application processes, using the Backur Naus form.

In practice, two applications do not exchange data about all their processes in one exchange. Only one or a set of processes at one time will exchange data. After identifying the concept to be exchanged, the steps outlined in the standard can be followed as :
the processes are defined and expressed using KIF syntax
the concepts contained in the processes (their names, relationship to other processes, conditional expressions) are further defined. In other words, the application's entities are given KIF definitions.
a translation is provided between the application's entities definition and PSL definitions.
At this point in the procedure, Applications A and B's processes have been expressed using PSL terms and KIF syntax. Each has a one-to-one correspondence between their process definition and a PSL definition. On this basis, data for the relevant process can be exchanged.

Following this procedure does not allow a software application to claim conformance to PSL according to ISO 18629, but it is sufficient for process exchange with another application. To this purpose, the National Institute of Standards has implemented a "question wizard" (PSL, Wizard) to facilitate the expression of any process with PSL definitions and in KIF syntax. A user specifies a process in details by answering questions and checking boxes for their process. The wizard returns a definition for the process using PSL.

7.3.3.2.3 User defined extensions
User defined extensions of PSL are extensions that introduce new primitive concepts. Typically, current extensions are sufficiently rich to express processes in existing software applications. However, the case where an application concept is not represented may arise. In this case, PSL can be extended to include a new extension by expressing it using the PSL Core, Outer Core, and definitions in existing extensions. The axioms in any extension that introduces new primitives must be consistent with the axioms of PSL-Core. User-defined extensions are needed when PSL is applied to domains that have not been yet dealt with in the extensions.

Research work has been done or is currently on-going, showing examples or interoperability among software tools using PSL, notably at the University of Stanford (CIFE) (Law 2001) (Cheng et al., 2003), and at the University of Loughborough (Cutting-Decelle et al., 2000), (Cutting-Decelle et al., 2001), (Cutting-Decelle et al., 2002), (Cutting-Decelle et al., 2004-2), (Tesfagaber et al., 2002). We present below an example of process exchange in construction using PSL.

7.3.3.2.4 Example of process exchange in construction using PSL

- Scenario : to illustrate interoperability for the construction domain, the following scenario has been designed:

The design and construction of an office building includes an exchange of information and data with the purpose of fitting a metal door to a metal wall frame. Conception, estimation of costs, and project planning must be studied for this scenario. Software applications used for this study include a design application using AutoCAD, a cost simulation software, and a project planner using MS-Project for the planning phase. A related scenario would be the exchange of processes for integrating a new supplier. Figure 7.17 illustrates this scenario. The process exchange is described in details for the AutoCAD and MS Project applications (Tesfagaber 2004).

[Insert Figure 7.17 here]

- Process interchange with PSL.

First the Architectural Design File is written using the syntax and terminology of the AutoCAD application. Second this file must be parsed to a KIF file, still using AutoCAD's terms and relations. This syntax translation may be executed with a parser between KIF and the AutoCAD's syntax. Thus an ontology for the design application is built. Third, the AutoCAD application's ontology is semantically translated into PSL terminology. This step is usually done manually or using the 20-question wizard developed at NIST for that purpose. This involves in-depth understanding of the processes in the AutoCAD application and may require consulting the documentation. It also introduces as many constraints and relations as possible on the AutoCAD terminology. Constraints and relations are taken from PSL and necessary for specifying in details what the AutoCAD terminology means. The result is a file where the AutoCAD is expressed using PSL concepts under specific conditions.

In parallel, the same process is performed for the process planning application using MS-Project terminology. Once both applications have been expressed in KIF and specified using PSL concepts, a inverse file containing constrained PSL concepts equivalent to the MS-Project application is created. Using these two files (Auto-CAD using PSL concepts and PSL concepts corresponding to MS-Project) process data can be exchanged under explicit conditions.

Below is the PSL translation of a process named door-assembly in AutoCAD developed using the 20-question wizard (Figure 7.18). This process is an activity. Its initiation depends on the state of other activities prior to this one (markov pre-conditions), but not on time or duration allocated to the activity. For instance, "make door frame" may be required for door frame assembly to occur. The result of the process is affected by the initial conditions existing prior to the process but not the duration. All occurrences of the "door assembly" activity have the same effect and are also time-independent.

[Insert Figure 7.18 here]

The specification of an MS task using PSL is given below. This specification intends to verify if a door-assembly process in AutoCAD can be equated to a task in MS project. If it was, only a syntactic parser for values of variables between AutoCAD and MS-project is necessary in information exchange. If not, the development of translation software may be assisted by providing an in-depth analysis of semantics and resolve the discrepancies using PSL.

In this example (Figure 7.19), a task is a constrained activity that cannot occur unless other activities have previously occurred: it depends on the state of another activity. This is similar to AutoCAD. However, an MS-task may also be bound by starting time, duration of the activity, or a combination of these with a state pre-condition. Therefore, if a parser is designed to translate from the AutoCAD file for door-assembly to a task in MS project, the parser must take into account the fact that time does not exist for the AutoCAD process so that the AutoCAD process is not equivalent to the MS task. In other words the parser can only partially input a new task based on the information provided by AutoCAD. Semantic encoding using PSL has here highlighted a potential source of error for automatic translators between two applications. In this case, it is determined that the AutoCAD process of door-assembly can be translated to an MS task constrained by the occurrence of activities, but additional constraints regarding time also exist. One possible solution to this obstacle is to enter by hand the values for time constraints, and duration.

[Insert Figure 7.19 here]

User defined extensions.

User defined extensions of PSL are extensions that introduce new primitive concepts, for instance for a domain where PSL has not been used before. Typically, current extensions and existing processes are sufficiently rich in PSL to express existing software applications. However, the case where an application concept is not represented may arise. In this case, PSL can be extended to include a new concept or extension by expressing it using the PSL Core, Outer Core, and definitions in existing extensions. The axioms in any extension that introduces new primitives must be consistent with the axioms of PSL-Core.

7.4. DE FACTO STANDARD DEVELOPED BY THE INTERNATIONAL ALLIANCE FOR INTEROPERABILITY (IAI): INDUCTRY FOUNDATION CLASSES (IFCs)

4.1 The International Alliance for Interoperability (IAI) Community

The International Alliance for Interoperability (IAI) is an international consortium of regional chapters registered and listed as non-for-profit organizations in North America, United Kingdom, Germany, France, Scandinavia, Japan, Singapore, Korea and Australia. Currently the IAI has about 650 membership organizations world-wide, being construction companies, engineering firms, building owners and operators, software companies, and academic institutions. The vision of the IAI is : "to provide a universal basis for process improvement and information sharing in the construction and facilities management industries.” (IAI 2001)

The vision is supported by the IAI mission statement : "to define, promote and publish the Industry Foundation Classes (IFC), a specification for sharing data throughout the project life-cycle, globally, across disciplines and across technical applications."
More information about the IAI is available at : http://www.iai-international.org.

7.4.2 The IFCs

The IFC are a data sharing specification, written in EXPRESS (10303-11, 1994), the dedicated formal language developed within the ISO 10303 STEP standard. Content according to IFC is currently exchanged between IFC compliant software applications using the Clear text encoding of the exchange structure, the STEP physical file (10303-21, 1994).

The scope of the IFC specification is the project life-cycle of construction facilities, including all phases as identified by generic process protocols for the construction and facilities management industries, such as : Demonstrating the need, Conception of need, Outline feasibility, Substantive feasibility study and outline financial authority, Outline conceptual design, Full conceptual design, Co-ordinated design, procurement and full financial authority, Production information, Construction, Operation and maintenance.
Development of IFC is guided by versions and releases, which do extend the scope successively. The processes supported by the current IFC2x specifications are : Outline conceptual design, Full conceptual design, Co-ordinated design, procurement and full financial authority, Production information, Construction, Operation and maintenance.
The target applications to exchange and share information according to IFC2x are : CAD Systems,

HVAC design systems, Electrical design systems, Formwork design and scheduling systems, Structural analysis systems, Energy simulation systems, Quantity take-off systems, Cost estimation systems, Production scheduling systems, Clash-detection systems, Product information providers, Steel and Timber frame construction systems, Prefab systems, stand-alone visualisation tools and others.

7.4.3 IFC Model Architecture

7.4.3.1 Architecture Principles

The IFC Model Architecture has been developed using a set of principles governing its organization and
structure. These principles focus on basic requirements and can be summarized as (IAI 2000) :
To provide a modular structure to the model.
To provide a framework for sharing information between different disciplines within the AEC/FM industry
To ease the continued maintenance and development of the model.
To enable information modelers to reuse model components
To enable software authors to reuse software components
To facilitate the provision of better upward compatibility between model releases

The IFC Model architecture provides a modular structure for the development of model components, the
'model schemata'. There are four conceptual layers within the architecture, which use a strict referencing
principle. Within each conceptual layer a set of model schemata are defined.

1. The first conceptual layer provides Resource classes used by classes in the higher levels.
2. The second conceptual layer provides a Core project model. This Core contains the Kernel and several Core Extensions.
3. The third conceptual layer provides a set of modules defining concepts or objects common across multiple application types or AEC industry domains. This is the Interoperability layer.
4. Finally, the fourth and highest layer in the IFC Model is the Domain layer. It provides set of modules tailored for specific AEC industry domain or application type.

The architecture operates on a 'gravity principle'. At any layer, a class may reference a class at the same or lower layer but may not reference a class from a higher layer. References within the same layer must be designed very carefully in order to maintain modularity in the model design. Inter-domain references at the Domain Models layer must be resolved through 'common concepts' defined in the Interoperability layer. If possible, references between modules at the Resource layer should be avoided in order to support the goal that each resource module is self-contained. However, there are some low level, general purpose resources, such as measurement and identification that are referenced by many other resources.

7.4.3.2 Gravity Principle : see Figure 7.20

1. Resource classes may only reference or use other Resources.
2. Core classes may reference other Core classes (subject to the limitations listed in 3) and may reference classes within the Resource layer without limitations. Core classes may not reference or use classes within the Interoperability or Domain layers.
3. Within the Core layer the 'gravity principle' also applies. Therefore, Kernel classes can be referenced or
used by classes in the Core Extensions but the reverse is not allowed. Kernel classes my not reference Core Extension classes.
4. Interoperability layer classes can reference classes in the Core or Resource layers, but not in the Domain layer.
5. Domain layer classes may reference any class in the Interoperability, Core and Resource layers.

[Insert Figure 7.20 here]

7.4.4 IFC Model Architecture Decomposition

The IFC Model Architecture for IFC 2x consists of the following layers.
Resource Layer
Core Layer
Kernel
Extensions
Interoperability Layer
Domain Layer

7.4.4.1 Resource Layer

Resources form the lowest layer in IFC Model Architecture and can be used or referenced by classes in the other layers. Resources can be characterized as general purpose or low level concepts or objects that are independent of application or domain need (that is, they are generally rather than specifically useful) but which rely on other classes in the model for their existence. For instance, geometry is a widely used resource whose specification is independent of domain. However, an object within a domain must be defined before its geometry can exist.
Exceptions to this characterization include classes from the Utility and Measure Resources that are used by other, higher-level resource classes.

7.4.4.2 Core Layer

The Core forms the next layer in IFC Model Architecture. Classes defined here can be referenced and specialized by all classes in the Interoperability and Domain layers. The Core layer provides the basic structure of the IFC object model and defines most general concepts that will be specialized by higher layers of the IFC object model (Figure 7.21).

[Insert Figure 7.21 here]

The Core includes two levels of generalization :

1. The Kernel : provides all the basic concepts required for IFC models within the scope of the current IFC Release. It also determines the model structure and decomposition. Concepts defined within the kernel are, necessarily, generalized to a high level. It also includes fundamental concepts concerning the provision of objects, relationships, type definitions, attributes and roles. The Kernel can be seen as a template model that defines the form in which all other schema within the model are developed (including all extension models). Its constructs are very general and are not AEC/FM specific, although they will only be used for AEC/FM purposes due to the specialization by Core Extensions. The Kernel constructs are a mandatory part of all IFC implementations. The Kernel is the foundation of the Core Model. Kernel classes may reference classes in the Resource layer but may not reference those in the other parts of the Core or in higher-level model layers.

2. Core Extensions : provide extension or specialization of concepts defined in the Kernel. They are the first refinement layer for abstract Kernel constructs. More specifically, they extend those constructs for use within the AEC/FM industry. Each Core Extension is a specialization of classes defined in the Kernel and develops further specialization of classes rooted in the IfcKernel. Additionally, primary relationships and roles are also defined within the Core Extensions. A class defined within a Core Extension may be used or referenced by classes defined in the Interoperability or Domain layers, but not by a class within the Kernel or in the Resource layer. References between Core Extensions have to be defined very carefully in a way that allows the selection of a singular Core Extension without destroying data integrity by invalid external references.

Goals for Core layer design are :

- definition of those concepts that are common to all parts of the model and that later can be refined and used by various interoperability and domain models
- pre-harmonization of domain models by providing the set of common concepts.
- stable definition of the object model foundation to support upgrade compatible IFC Releases

7.4.4.3 Interoperability Layer

The main goal in the design of Interoperability Layer is the provision of schemata that define concepts (or classes) common to two or more domain models. These schemata enable interoperability between different domain models. It is at this layer that the idea of a 'plug-in' model approach emerges. It is through the schemata defined at the Interoperability Layer that multiple domain models can be plugged into the common IFC Core. The 'plug-in' approach also supports outsourcing of the development of domain models.

7.4.4.4 Domain Layer

Domain Models provide further model detail within the scope requirements for an AEC/FM domain process or a type of application. Each is a separate model that may use or reference any class defined in the Core and Independent Resource layers. Examples of Domain Models are Architecture, HVAC, FM, Structural Engineering etc. An important purpose of Domain Models is to provide the 'leaf node' classes that enable information from external property sets to be attached appropriately.

7.4.5 Connecting External Models to the IFC Model

Fully harmonized IFC Domain Models are directly connected the Core definitions. Domain Models that are not fully harmonized have to provide appropriate connection to relevant IFC class definitions in order to use the IFC model framework. Such models may be developed according to different technical architectures and methodologies but might need to be used in conjunction with the IFC model at some point.
The means of achieving this is through the use of a connection mechanism. The main requirements for connection are the facilitation of :

1. Connection of externally developed, non harmonized, Domain Models via a connection that provides a mapping mechanism down to Core and Interoperability definitions. The definition of the connection is in the responsibility of the Domain Model developer and is part of the Domain Model Layer.
2. Establish an inter-domain exchange mechanism above the Core to enable interoperability across domains. This includes a container mechanism to package information. Therefore a connection is used where the definition of the connection is the responsibility of all Domain Models that share its use.

Connections are based on Core Extension definitions and enhance those Core Extension definitions. Those enhancements provide common concepts for all Domain Models that might further refine these concepts. As an example, the Building Element provides the definition of a common wall, whereas the Architectural Domain Model will enhance this common wall with its private subtypes and type definitions. A connection that is used by several Domain Models therefore provides a level of interoperability through shared connection definitions.
Non-IFC harmonized models can be connected to the IFC Core Model through a specifically defined mapping. For specific high-level inter-domain exchange that cannot be satisfied by common definitions in the Core, connection through mapping may provide a specific inter-domain exchange capability.

7.4.6 Overall Architecture

The following diagram (Figure 7.22) shows the complete set of IFC 2x model schema organized according to the layer at which they exist.

[Insert Figure 7.22 here]

Note that all schema are named in a manner that enables identification of their architecture layer :

Schema at the resource layer are suffixed with the term ‘Resource’
Schema at the core extension layer are suffixed with the term ‘Extension’ (other than the Kernel schema which is considered to be a special case)
Schema at the interoperability layer are suffixed with the term ‘Elements’
Schema at the domain layer are suffixed with the term ‘Domain’

Industry Foundations Classes IFC2x have been endorsed by the ISO organisation as the ISO/PAS 16793 in November 2002 (IAI 2001).

7.5. CONCLUSION: CONTRIBUTION OF DIFFERENT STANDARDS TO INTEROPERABILITY IN CONSTRUCTION

We have shown to what extent standards-based approaches can be helpful to facilitate information sharing and interoperability among software applications commonly used in manufacturing, and in manufacturing management. Most of the time, technical terms handled by those applications look similar, or, even worse, are exactly the same – however their meaning is different.

In the four standards described above, technical terms are established more or less on the same “construction-flavoured” vocabulary, but are very different,with multiple interpretations of the same terms in each standard. Given its properties, and its structure, ISO 18629 PSL can be considered as a powerful interoperability “tool” for the information systems of the enterprises. It introduces economy of scales – each application only needs to provide interoperability to PSL once for information exchange (Figures 7.23 and 7.24). If an application changes, it is up to the developers of this application to provide new translations to PSL. Thus only one application in the chain of inter-operation is affected and not the others.

[Insert Figures 7.23 and 7.24 here]

However, implementation of standards is non-trivial and costly. It is not until the use and implementation of these standards in a particular industry has reached “critical mass” that costs will decrease. STEP already has made great strides in this direction. The construction industry may benefit from the lessons learned in other domains, particularly for Concurrent engineering based approaches of construction projects (see chapter 4).

7.6. ACKNOWLEDGEMENT

The authors wish to thank G. Tesfagaber, PhD student and Prof. N.M. Bouchlaghem, Supervisor of her research work at the University of Loughborough.

7.7. REFERENCES

Anumba C.J., Cutting-Decelle A.F., Baldwin A.N., Dufau J., Mommessin M., Bouchlaghem N.M., Introduction of Concurrent Engineering concepts into an integrated Product and Process Model, Concurrent Engineering in Construction Conference, 1999

Cheng J., Gruninger M., Sriram R.D., Law, K.H., Process specification language for project scheduling information exchange, International Journal of IT in AEC, Vol. 1, Issue 4, Dec 2003

Cutting-Decelle A.F., Michel J.J., Schlenoff C., Manufacturing and construction common process representation : the PSL approach, CE00 Lyon 2000

Cutting-Decelle A.F., Michel J.J., Schlenoff C., Integration of industrial management information and interoperability : MANDATE + PSL a combined approach, MICAD 2000.

Cutting-Decelle A.F., Anumba C.J., Baldwin, A.N., Bouchlaghem N.M., Tesfagaber G., PSL : a common language for the representation and exchange of process information in construction, 1st International Conference on Innovation in AEC, CICE, University of Loughborough, UK, July 2001

Cutting-Decelle, A.F., Dubois, J.E., "Heterogeneous information exchanges in the construction industry – the SHYDOX/MATCOMP/Xi prototype", in Heterogeneous information exchange and organizational hubs, Bestougeff, Dubois and Thuraisingham Eds, Kluwer Academic Publishers, 2002

Cutting-Decelle A.F., Anumba C.J., Baldwin, A.N., Bouchlaghem N.M., Tesfagaber G., Exchanges of process information between software tools in construction : the PSL language, International Conference ECPPM02, Portoroz, Slovenia, Sept 2002

Cutting-Decelle A F, Pouchard L C, Das B P, Young R I, Michel J J, "Utilising standards based approaches to information sharing and interoperability in manufacturing decision support", International Conference FAIM04, Toronto (Canada), July 2004

Cutting-Decelle A.F., Young R.I.M., Das B.P., Anumba C.J., Baldwin A.N., Bouchlaghem N.M., "A multi-disciplinary representation of the supply chain information in construction : an innvoative approach to project management", TMCE Conference, 2004

Etchemendy J., The language of first-order logic, 3rd Edition, CSLI Lecture Notes N° 34, 1992

Fikes, R. and A. Farquahr. 1999. "Distributed Repositories of Highly Expressive Reusable Ontologies." IEEE Intelligent Systems and their Applications 14, no.2 (March-Apr.)

FOLDOC, Free On-Line Dictionary of Computing. Imperial College Department of Computing. http://wombat.doc.ic.ac.uk/foldoc/

Genesereth, M. and R.E. Fikes. 1992. "Knowledge Interchange Format, Version 3.0. Reference Manual.", Technical Report Logic-92-1. Computer Science Department, Stanford University, Stanford, CA. (Jan.).

Gomez-Perez, A. 1998. "Knowledge Sharing and Re-Use." In The Handbook of Applied Expert Systems, J. Liebowitz, ed. Boca Raton, 10:1-10:36.

Gruber, T. 1993. "A Translation Approach to Portable Ontology Specifications." Knowledge Acquisitions 5, (May): 199-220

Gruninger, M., A guide to the ontology of the Process Specification Language, in "Hand-book on Ontologies in Information Systems", R. Studer and S. Staab Eds, Springer-Verlag, 2003

http://www.iso.ch (ISO Web site)

http://www.mel.nist.gov/psl (PSL Web site)

http://www.mel.nist.gov/psl/20questions.html) (PSL Wizard)

http://www.tc184-sc4.org (ISO TC 184 / SC4 Web site) http://philebus.tamu.edu/cl/ (Common Logic Web site, 2004)

http://www.whatis.com

http://www.ariadne.ac.uk/issue24/interoperability/intro.html, (Interoperability, 2000)

http://www.nist.gov/sc5/soap/ (SOAP, Step On A Page)

Hunhs, M. N. and M. P. Singh. 1997. "Ontologies for Agents." IEEE-Internet Computing 1, no. 6 (Nov.-Dec. 1997)

IAI, IFC release 2x, IFC technical Guide, 2000

IAI, Transposing the IFC2x Specification as ISO Standard, 2001

ISO 10303-1, "Industrial automation systems and integration – product data representation and exchange – part 1 : overview and fundamental principles", 1994

ISO 10303-11, "Industrial automation systems and integration – product data representation and exchange – part 11 : the EXPRESS language reference manual", 2003

ISO 10303-21, "Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: "Clear text encoding of the exchange structure", 1994

ISO 10303-22, "Industrial automation systems and integration – Product data representation and exchange – Part 21: Standard data access interface", 1998

ISO 10303-23 , Industrial automation systems and integration – Product data representation and exchange – Part 23: Implementation method: C++ language binding to the standard data access interface, 2000

ISO10303-24, Industrial automation systems and integration – Product data representation and exchange – Part 24: Implementation method: C language binding to the standard data access interface, 2001

ISO10303-26, Industrial automation systems and integration – Product data representation and exchange – Part 26: Implementation method: Interface definition language binding to the standard data access interface, 1997

ISO 13584-1, Industrial automation systems and integration – Parts Library : Overview and fundamental principles, 1999

ISO 13584-42: Industrial Automation Systems and Integration, Parts Library : Methodology for Structuring Parts Families, 1998.

ISO IS 15531-1, "Industrial automation systems and integration – Industrial manufacturing management data – part 1 : general overview", 2002

ISO IS 18629-1, "Industrial automation systems and integration – Process specification language – Part 1: Overview and basic principles", 2004

ISO IS 18629-11, "Industrial automation systems and integration – Process specification language – Part 11: PSL-Core", 2004

ISO IS 18629-12, "Industrial automation systems and integration – Process specification language – Part 12: Outer Core", 2004

ISO DIS 18629-13, "Industrial automation systems and integration – Process specification language – Part 13: Duration and ordering theories", 2004

ISO DIS 18629-14, "Industrial automation systems and integration – Process specification language – Part 14: Resource theories", 2004

ISO DIS 18629-41, "Industrial automation systems and integration – Process specification language – Definitional extensions: Part 41 : Activity extensions", 2004

ISO DIS 18629-42, "Industrial automation systems and integration – Process specification language – Definitional extensions: Part 42 : Temporal and state extensions", 2004

ISO DIS 18629-43, "Industrial automation systems and integration – Process specification language – Definitional extensions: Part 43 : Activity ordering and duration extensions", 2004

ISO CD 18629-44, "Industrial automation systems and integration – Process specification language – Definitional extensions: Part 44 : Resource extensions", 2004

Kemmerer, S. J., The grand experience, NIST SP 939, 1999

Knowledge Interchange Format, Part 1 : KIF-Core, ISO/JTC1/SC32/WG2, WD, 1999

Law, K.H., "Process specification and simulation",PSL quaterly progress report, Stanford University, CIFE, 2001

NIST: " Interoperability Cost Analysis of the U.S. Automotive Supply Chain (Planning Report #99-1)", 1999. Available at http://www.nist.gov/director/prog-ofc/report99-1.pdf

NIST,"Economic impact assessment of the international standard for the exchange of product model data (STEP) in transportation equipment industries", Planning report #02-5, 2002

Pierra G. and Sardet E., Potier J. C., Battier G. and Derouet J. C., Willmann N. and Mahir A., "Exchange of component data : the PLIB (ISO 13584) model, standard and tools", Proceedings of the CALS EUROPE'98 Conference, 16-18 September 1998, Paris, France.

Pierra G., Dehainsala H., Ait-Ameur Y., Bellatreche L., Chohon J., El-Hadj Mimoune M., "*Base de Données à Base Ontologique : le modèle OntoDB*", Rapport de recherche 2004-02 (soumis à publication), 2004.

Pouchard, L., Ivezic N., Schlenoff, C., Ontology engineering for distributed collaboration in manufacturing, AIS, 2000

Pouchard, L., Rana, O., "The Role of Ontologies in Agent-oriented Systems", 6th Joint Conference on Information Sciences (JCIS), Computational Semiotics Workshop, Research Triangle Park, NC, March 2002.

Ray S. R. and Jones A. T.: "Manufacturing interoperability", Concurrent Engineering: Enhanced Interoperable System: Proceedings of the 10th ISPE International Conference, pp.535-540, 2003.

Tesfagaber G., Cutting-Decelle A.F., Anumba C.J., Baldwin A.N., Bouchlaghem N.M., "Semantic process modelling for applications integration in AEC", International workshop on information technology in civil Engineering, ASCE, 2002