

Chapter 5

How to use PSLX specifications

Stakeholders in the standard

The PSLX specifications are used as a tool for implementation of management systems in accordance with the APS of manufacturing industries. Furthermore, the specifications act as a form of common rules to provide greater benefits for all stakeholders. The expected benefits of applying the PSLX specifications can be listed as follows:

- (1) Support for realizing successful grand design for APS implementation
- (2) Support of system life-cycle management and evolutional development
- (3) Establishment of interoperability between different applications
- (4) Realization of system extensibility and environment for add-in software
- (5) Facilitation of user-oriented system development and continuous improvement

(6) Sustainable support for business data management and re-design processes

The impact of these outcomes of the specifications differs according to the stakeholder. Advantages to respective stakeholders are:

(1) CEOs of manufacturing businesses

For CEOs or executive personnel in manufacturing enterprises, the specifications provide detailed information on the competitive advantages and functionalities of an APS. The information depends on technical background, so that the recommendation is not some kind of sales pitch, but rather a reliable guide to decision-making. Investment in an enterprise information system is significant and can have a huge impact on the future of the enterprise. The specifications guide the directors in developing a clear vision and grand design for the enterprise.

(2) Business process designers

For business process designers who understand and improve the detailed business processes in a manufacturing enterprise, the specifications point the way to a clearer definition of their processes, facilitating discussion of such knowledge with a broader range of people in the enterprise. This converts implicit knowledge of the processes into explicit information to facilitate collaborative design of a business process between different business divisions. Moreover, it becomes possible to efficiently manage data used across several business activities by clarifying the primary division of business activity that will be responsible for managing the life cycle of the data, from creation through revision, storage and disposal.

(3) Information system designers

First, information system designers can make decisions as to the priority of any system developments in the enterprise with respect to the total balance to be achieved in the business model. Some systems may need

to be designed from scratch, while others can be developed by modifying legacy systems. Using the specifications, the designer can analyze and select best-of-breed software or IT service providers, comparing in detail the requirements of the enterprise and available external solutions.

(4) IT consultants/business consultants

For business consultants/IT consultants specializing in manufacturing industries, the specifications provide a fundamental knowledge environment in which to initiate discussions with their manufacturing clients. Common terminologies and shared understanding of business activities on plant floors facilitate progression to more detailed and company-specific discussions. Furthermore, when they seek to implement a formal analysis of enterprises, or benchmarking, the specifications enable them to evaluate a number of quantitative factors by focusing on the detailed structure of the business model, instead of basic and largely conceptual qualitative factors.

(5) Systems integrators

Systems integrators, who deal with numerous different software applications to arrive at solutions for particular problems in different industries, gain an advantage from the specifications by being able to make significant reductions in the labor needed to achieve interoperability between systems. Such interoperability is one of the most important issues in PSLX; accordingly, a number of technologies for interoperability are proposed in the specifications. The business approach of systems integration companies will become more knowledge-intensive and value-added.

(6) IT package vendors

For IT package vendors who provide software packages as best-of-breed IT solutions in their specialized application fields, the PSLX specifications can expand business opportunities and market size. If they develop an

interface of PSLX common schema, the software can connect to third party products. As the number of the applicable third party software increases, business also increases. Small and medium-size manufacturers emerge as a new market for their services.

The remainder of this chapter describes some typical cases in which the above stakeholders are trialing PSLX specifications in their specific contexts in real-world business situations.

Engineering for current business processes

This section introduces the case for considering the grand design of a manufacturing enterprise in terms of the fundamental structure of decision-making systems specifically related to planning and scheduling in accordance with the PSLX specifications. It looks at redesign of the business model with suitable business processes and reallocation of decision-making modules, appropriate to the current categories of business models in particular business environments. Appreciation of best practice in this field will aid understanding of this topic.

In designing a business model for enterprise decision-making, the planning horizon and the planning cycle are important factors in each decision layer. This is the first step in determining an overall decision-making structure. The collaboration patterns of planning and scheduling should also be designed in advance, while the granularity of planning decisions is decreasing. All decision-making systems for all business processes in the enterprise are based not only on well-formed computer systems, but also on people-centered business decision systems, wherein both human and computer interactions can capture real and precise business information.

The business processes so designed need to be described in terms of use cases and collaborations between particular business activities. In

design processes, business activities are connected to create a flow of information, which results in a business process. First, both the “to-be” model, which is the final goal of the enterprise, and the “as-is” model, which is the current status of the enterprise, are described and compared. Then the designer should identify an intermediate stage of the business model that is possible to achieve early in the system development process. A step-by-step future development plan is also necessary in order to achieve the final stage defined as the to-be model for the enterprise.

One of main features of APS defined by PSLX is an operational level collaboration between business activities using scheduling software applications. This includes not only horizontal collaboration between different area schedulers on plant floors, but also vertical collaboration between the plant floors and business planning divisions, which deal with more aggregated but wide-ranging information. Scheduling in business divisions is centralized decision-making, whereas scheduling on plant floors represents decentralized and distributed systems. Vertical collaboration between these different systems involves such a variety of patterns so that it requires the support of specialized PSLX knowledge.

Conventionally, the systems knowledge of a business process and that of an engineering or manufacturing process are divided into two individual systems. However, PSLX discusses those two separate worlds on a single APS system platform. The APS domain model defined by PSLX allows the designer to represent both business process knowledge and engineering knowledge at the same time, because there is no distinction between the two. This means that PSLX can integrate the interdisciplinary area between engineering and business as a practical and value-adding application field of information technologies.

Profiling the various software systems

Information systems in current manufacturing enterprises are very complex and difficult to implement for information systems support personnel. Even an IT solution vendor cannot keep abreast of all the technologies that need to be applied in an enterprise-wide system development project. Therefore, the best-of-breed approach, in which many IT solutions are combined in a complete system, performs very well in such a diversified environment. However, there is a problem with this approach, in that many kinds of software feature a huge variety of functions, which are difficult to understand in the context of each manufacturing enterprise unless the user invests considerable time and money in actually buying and evaluating the system. As a general rule, finding appropriate software is a difficult and potentially risky process.

The PSLX specification for defined business activities is used as a software profiling tool that shows software functions and application fields in detail. This is valuable for manufacturers when soliciting proposals from IT vendors or systems integrators. It also benefits IT package vendors when promoting their products, and systems integrators when proposing systems to their clients. The procedure of system profiling is as follows.

First, all the business activities defined in the PSLX specification are checked to determine whether or not the target software covers the relevant functions. The corresponding areas of business activities are usually adjacent to each other within the same view of administrative or functional classifications. If the software contains several independent areas on the PSLX business activity model, then it needs to be divided according to those areas, so as to achieve clear definition in the software.

The second step is to elaborate use cases registered in the business activities in the target area of the software. If there is no use case that represents the particular requirement, a new use case should be described in the same representation format. The new case can be

submitted for registration as a PSLX common use case. The third step involves describing business collaboration by referring to the PSLX specification. This has some similarity to Step 2, in that the corresponding specifications are selected and elaborated for the system.

Finally, the system interfaces to other business activities that are not a part of the system but are within a same collaboration defined by PSLX should be described as part of its functional specifications. Specification of the system interfaces is also required if a collaboration is within a business activity but there are several distributed software modules for the same business activity across the target manufacturer's operations.

Data management and continuous improvement

In typical enterprise information system development projects, the process that consumes the greatest time and effort is practical data preparation for regular execution of all business processes in accordance with the information system. This process is more complex than a specific software module customization for each requirement because it requires the involvement of end users in creating various new data. PSLX specifications show the divisions responsible for each set of data, while business activities in the PSLX specification are mapped to particular software functions, and data items necessary for the activities are clarified. Therefore, it is possible to start preparing data for the new information system before development has been completed.

When implementation of a new business process is planned for competitive advantage purposes, the cost of gathering data to maintain the process is an important factor, as is the cost of software module development. Even if a completely new software system suite is chosen, most data managed in the system suite are of a conventional type, collected and modified from the current business activity. Data migration support thus becomes a very important process in system development.

This is a kind of business data centric approach to software system development for particular systems that involve a significant number of human decision activities.

An example of reverse engineering of a legacy system is used to explain a case of application of the PSLX specification. First, similar to the process described earlier in the section, “Profiling the various software systems”, the functions and behaviors of the legacy system are described in detail, together with the business activities and collaborations related to the system. Then, every transaction performed in the use cases and collaborations is described in terms of data classifications by means of the APS domain model. This step does not require data contents, but does need a list of classes.

Reverse engineering processes, in which RDB schemas in the legacy system are analyzed in detail, usually entail onerous work involving considerable time and effort. In many cases, schemas in the legacy systems are not well formed and easy to understand, due to histories of multiple modifications and use of intuitive terminologies. The process of analyzing the detail of implementation schema is time-consuming and non-productive. In its place, PSLX offers a more elegant and sensible approach that uses only the external specifications of the legacy system in order to obtain an abstract data model.

After arriving at an abstract data model corresponding to real-world business activities, the next step is to develop a particular implementation schema, and then input various data captured from actual situations. In the first few weeks, it may be necessary to have duplicate data input to both the legacy system and the new system. In some cases, package vendors provide data migration tools; however, it is preferable to develop an independent system for data preparation. Even though many IT system development projects focus on the ability of processes to introduce new software functions, frequently the issue of transforming business processes from current real-world operations to the new design is much more important and critical in terms of the ultimate success of the

enterprise.

Another example of application of the PSLX specification in terms of application data consistency management can be shown in a case of bill of materials (BOM) management. Depending on the particular manufacturing business context, there may be several types of bills, such as engineering BOM, manufacturing BOM, sales BOM, and service BOM. This different BOM information is usually stored in different RDBs and managed separately, even though the data separately stored represents the same information. Therefore, data consistency management for the distributed databases is required.

Data consistency management is difficult if the system architecture is not on a centralized basis. Many distributed and autonomous systems have disadvantages in terms of data consistency. To solve this problem, the PSLX specifications provide a guide to maintaining consistency between the different data implementations in distributed RDBs by addressing an APS domain model or PSLX ontology as common identifiers.

In order to maintain data consistency, some rules need to be prescribed. For example, if the same information is allocated on two different databases and both are rewritable, then a negotiation rule that defines a process for revising the data according to the other database must be prescribed. This rule can achieve synchronization between these distributed databases. This is possible because the PSLX specification can show semantic identity between different data fields in different data tables, in different RDB implementations.

Support for RDB schema design

PSLX information system architecture is distributed and autonomous architecture rather than centralized. Centralized architecture that manages data in the same place has a long history in data processing -

from the mainframe computer era to the client server information system era. On the other hand, distributed and autonomous architecture that consists of small databases is a new concept, and not so popular compared to the previous type. However, the advantage of this architecture is its ability to process extensional and precise data in a formal data format, and flexibility to adapt to changes in the data schema that closely correspond to continuous improvement. The effect of revision on all business processes can be minimized.

In order to get this advantage, many people will have to learn the techniques of schema design so that adjustments to their own localized business environment can be made. Schema design hitherto has tended to be a special knowledge preserve of information system engineers. System implementation techniques for proprietary systems are outside this scope; however, a particular business environment has to be covered by a data schema that has sufficient data and can easily be applied by efficient business algorithms. This is demanding work for a typical IT engineer.

The PSLX-RDB common schema is a template for actual database schemas that meet each requirement of a real-world business environment. Using the PSLX specifications, an average information system designer can create a high-quality RDB schema without any special knowledge of schema design. The procedure to define schemas is outlined below.

First, suitable classes in the PSLX-RDB common schema are selected for each business activity, taking into account the particular use cases involved. If special use cases not included in the PSLX specification are required, then the minimum set of classes containing all information in the use case is added. All the classes selected are defined as views in the target RDB schema.

The classes in a PSLX-RDB common schema each have a minimum set of attributes, where an attribute corresponds to a data field in a view of the

RDB. Data in the enterprise business activities are partially represented by the attributes; however, most data require definition of an additional attribute in a class of PSLX-RDB common schema. When a new class or attribute of a class is added to the implementation schema, the position of the additional object in the APS domain model needs to be clarified. Then, the semantics of the additional information can be distinguished, so as to allocate the same data to several different classes in the PSLX-RDB common schema.

The next step is divided into two cases: designing a new RDB schema from scratch, and designing additional classes and modifying some classes in a legacy RDB schema. For new design cases: First, all the classes that are in the APS domain model and may have an attribute of the common RDB schema are listed. The classes listed there are defined as tables in the implementation schema. Furthermore, additional tables for relations between the classes are defined with respect to cardinality. If the relations correspond to relation classes in the PSLX-RDB common schema, they are also added to the table list.

Consequently, a new implementation schema is defined based on the APS domain model, by a schema-transforming procedure in the PSLX specification. This procedure allows the schema to move or duplicate a field between different tables in order to increase calculation speed and reduce the complexity of links between tables. Finally, the data types of each field, i.e. an attribute of a class, are determined. The implementation schema completed to this point should be able to generate a subset of the PSLX-RDB common schema as a view of the tables by SQL or stored procedures. It is possible to create an implementation schema based directly on the PSLX ontology, rather than on the APS domain model.

On the other hand, if there is a legacy RDB system to be modified so as to maintain the data, the schema of the current RDB needs to be associated with the PSLX-RDB common schema by providing corresponding views. In the first step, the schema of the legacy RDB is mapped on the PSLX-RDB common schema. Then, similar to the case of new schema

design, classes in the APS domain model are selected, taking into account necessary relations between the classes in the PSLX-RDB common schema and the classes in the APS domain model. Consequently, the additional data object that exists in the APS domain model, but not in the implemented RDB schema, is identified. If the data object is an attribute of an existing class, then it is described as an attribute of a class in the implementation schema that initially represents the legacy schema. If the data object is a new class, then it needs to be added to the implementation schema.

The procedure to produce a final implementation schema contains further, subsequent detailed steps. These involve precise operations peculiar to individual system environments, so they have been omitted from this paper. In conclusion, the common thread in the two cases of implementation schema design is that the PSLX-RDB common schema is used as a common interface for access to internal data from outside the application software. The data schema inside the implementation schema can be revised frequently without any great concern about inconsistency with other data in different RDBs, while the semantic relations are maintained in the common schema for all the data in the implemented RDB, so that the detailed data schema in each application does not affect those of the others.

Application integration in manufacturing

Systems integration is very demanding work in an information system development project, and is usually where the major technical difficulties arise. Systems integration requires suitable communication capability between two programs. It also requires methods to access a common database. But, that's not all. The most important effort is the one that goes into integration of two or more business processes managed by different sub-systems so that they will perform as a whole, taking into account the common objective.

In reality, it is very difficult to make even the first two aspects of integration work properly. If the two programs that need to communicate with each other are designed by the same designer or the same IT vendor, with agreement between any communication protocols, there should be no problem. However, if the system does not know its future communication partners, and if they should be designed by different parties, there will be a question mark over the likelihood of suitable communication. Furthermore, even in a case where two different programs try to connect to a common database, the detailed information of the database schema is sometimes not available to read, or the information usually does not have any semantics, so that the program cannot find the correct data in the database.

This represents a kind of design problem, as well as a communication problem automatically produced by computers. In PSLX specifications, the semantics of each field of databases can be defined using PSLX ontology. Especially on plant floors, there can be a wide variety of terminologies depending on different company cultures, different divisions and personnel, even within a single enterprise. If a term used in each situation differs on the plant floor, PSLX ontology can locate the same information object by eliminating the ambiguity of the original definition. Furthermore, the APS domain model provides a standard terminology set, which can be useful in sharing information between different organizations without the need to translate using an ontology.

It is impossible to implement a system with full knowledge of all the requirements that will be arise in future communication with currently unidentified future partners. In such a case, the best approach is to define a common schema and protocol whereby all systems that wish to communicate with others support the interface. This involves legacy systems, as well as systems planned for future development. The PSLX-RDB common schema and the PSLX-XML common schema are really examples of this approach. Therefore, a system that has an interface of PSLX-RDB or PSLX-XML can communicate with other

systems without any additional input.

Interface design for flexibility, as a component of large systems, is an important issue in developing an IT software package. For this purpose, the PSLX specification helps the package designer in defining the specification of the software. First, the target business activities are selected from the whole set of PSLX business activities. Then, collaborations supported by the package are decided. Third, a communication type is chosen; either a storage type communication by RDB, or a messaging type communication by XML technologies. Then, the appropriate part of the PSLX-RDB common schema or PSLX-XML common schema is selected. Finally, the system has an interface to accept immediate access from other systems, and publishes the information of the interface.

The case where a system only has the capability to access other systems that have PSLX interfaces when it needs to retrieve information is very easy to implement, because there is no need to constantly prepare unknown accesses. In this case, the system needs to have a PSLX-RDB or PSLX-XML common schema, and to only translate data between the internal format and the schema. This is very useful for many kinds of application software that do not have a special database, but instead have special algorithms for particular problems. They can be provided as plug-in software components. There are many personal plug-in business software packages, but only a few software packages in business applications for enterprises. Industrial business knowledge itself can be marketed under this approach in the future.