Integration of product life cycle views on the basis of a shared conceptual information model

Prof. Dr.-Ing. Reiner Anderl

Dipl.-Inform. Anne Wasmer

FG Datenverarbeitung in der Konstruktion (DiK)

FB Maschinenbau, TH Darmstadt

D-64287 Darmstadt, Petersenstraße 30

Telephone: 06151/16-6001, Fax: 06151/16-6854

email: anderl@dik.maschinenbau.th-darmstadt.de

wasmer@dik.maschinenbau.th-darmstadt.de

Abstract

The paper presents the development and modelling approach of ISO 10303 ('Product Data Representation and Exchange') with a focus on the mapping approach currently taken. As result of the mapping process, the so-called application interpreted model (AIM) is created that is dedicated to be used as a basis for implementations. The paper describes an approach how the AIM schema which is the target of the mapping process and which forms the basis for integration can be automatically derived and how this approach then can be combined with existing mapping languages suggested in the STEP arena with the goal to integrate application specific life cycle views in an enterprise.

Keywords

application protocol, unit of functionality, mapping, mapping table, mapping language, view

1 INTRODUCTION

Due to the increasing need for computer systems support during all phases of the product life cycle, a lot of computer aided systems have been developed and been used during the last years. They are usually specialized to support a specific application and are designed based on their own individual information model representing their particular application domain view on the product and the product model respectively. As a consequence, these application systems build

islands of automation and as such don't allow for the joint use of the product data described by those information models. The result is on one hand a wealth of information models tailored to meet specific application requirements and overlap in the semantic contents they are representing, and on the other hand the redundancy of product data stored in the different application systems which makes it difficult to maintain the product data and keep them consistent during the product life cycle. Nowadays, new working methods and techniques like total quality management, concurrent and simultaneous engineering, together with an increasing need for co-operation between enterprises already during the development and design process require continuous process chains with an integrated information technology support throughout the whole product life cycle. Product data technology plays a key role in this context. The development of ISO 10303 (STEP) is an example for the product data technology approach to create an integrated information model representing all information necessary to describe a product throughout its whole life cycle. As the purpose of this model is to support individual applications or process chains, the concept of the so-called application protocols has been introduced. An application protocol (AP) in ISO 10303 defines all information needed to support a particular process chain or application. The definition of an 'application' to be 'a group of one or more processes creating or using product data' (ISO 10303-1, 1994) has led to the definition of application protocols with a significant difference in the size of their scope. While some APs define their scope as being the context for one computer application independently from a certain type of product, e.g. AP202 'Associative Draughting' (ISO 10303-202, 1996), others deal with complete process chains involving a lot of different computer applications, e.g. AP214 'Core data for automotive mechanical design processes' (ISO TC184, 1995-2), that covers the development of automotive products including the definition of requirements, styling, design, evaluation, production planning of the automotive parts as well as design and manufacturing of the tools used to produce them. Depending on the scopes of the existing APs, computer applications used during the life cycle of a product can be integrated on basis of one single AP or multiple APs can be combined to cover the information requirements of the computer applications to be integrated. In the following, it is shown how the three layer architecture of APs can be combined with existing mapping languages suggested in the STEP arena and be used as the basis to integrate specific views of computer applications throughout a process chain.

2 ARCHITECTURE OF ISO 10303 APPLICATION PROTOCOLS

The objective of ISO 10303 is to provide a neutral mechanism capable of describing product data throughout the whole life cycle of a product, independent of any particular application system. Although ISO 10303 has been designed to support the product data exchange, its nature makes it also suitable as a basis for implementing shared product databases and archiving. While product data exchange is defined as the transfer of product data between a pair of applications that operate upon their own copies of the product data, product data sharing is characterized by more than one application logically operating on a single copy of the same product data (ISO TC184, 1996-2).

ISO 10303 is organized in different series including description methods (10 series), containing e.g. the formal information modelling language EXPRESS (ISO 10303-11), implementation methods (20 series), methods for conformance testing (30 series), and the product data specifications that are divided in three categories: integrated resources (40 and 100

series), application protocols (200 series), and application interpreted constructs (500 series).

The **integrated resources** are a coherent set of modularly structured schemas specified in EXPRESS. They can be seen as an integrated conceptual reference model for product data. They are defined independently of a specific application context in which product data might be used. The extension of the Integrated Resource models is done in a synergistic process following the integration methodology described in (ISO TC184, 1996-2). Its purpose is to maintain the Integrated Resources as a single, self-consistent model (consisting of different schemas) free of redundancy.

Application protocols are defined for a specific industrial application domain or application context specifying the functional aspect of the information, i.e. the collection of processes that is supported by the application protocol, the types of product or industry sector, the life cycle stage and discipline. An application protocol is based on a three layer architecture and consists of three types of models described in the following.

Figure 1 Components of an Application Protocol.

- The **application activity model (AAM)** specifies the enterprise activities ('what is done?' versus 'how is it done?'), that are constituting the application context, and the information flows between them in IDEF0.

- The **application reference model (ARM)** details the information flows identified in the AAM. It specifies the information requirements of the application protocol, i.e. for the defined application context, in the terminology of the application domain and structured in the way the domain experts think of it. It is graphically represented as an information model in EXPRESS-G, IDEF1X or NIAM together with a textual definition of the identified so-called application objects, i.e. entities and their attributes (application objects), and their relationships (application assertions).
- The **application interpreted model (AIM)** is based upon the semantic content of the ARM. The AIM is specified in EXPRESS and used as a basis for implementations of product data exchange and also forms the conceptual basis for other implementations like shared databases. It consists of Integrated Resource constructs and specializations of them that semantically correspond to the application domain's information requirements (described in the ARM) and of constraints to restrict the population of the AIM according to the means of the particular application domain. These correspondences are identified during the so-called interpretation process (mapping) and documented in the mapping tables.

When two or more application protocols have an overlap in their information requirements, they have to use the same interpretation, i.e. the same set of Integrated Resource constructs and their specializations to fulfil these requirements in the AIM. In this case, the set of common AIM constructs is specified in a separate EXPRESS schema, the **application interpreted construct (AIC)**. All application protocols which have been identified as sharing a common information requirement use the same application interpreted construct in its entirety.

3 METHODOLOGY OF ISO 10303 APPLICATION PROTOCOL DEVELOPMENT

After defining the textual definition of its scope and identifying the application context, the application protocol is developed in phases that comprise AAM, ARM, mapping table, and AIM development. The phases of AP development are described in detail in (ISO TC184, 1996-2). In the following, the main focus is on those concepts that are considered to be essential from an application integration perspective. To illustrate the concepts, examples from AP214 (ISO TC184, 1995-2) are provided.

To handle the complexity and to provide a mechanism for modularizing the AP on the requirements definition level, ARMs are structured in so-called Units of Functionality (UoF). A unit of functionality is a collection of entities, their attributes, and relationships, that conveys one or more well-defined concepts within the context of an AP's ARM. The ARM constructs grouped into a UoF build a logical unit that usually supports an application function or process. Modularizing the ARM into units of functionality facilitates on one hand the development of the ARM itself as a clearly structured model without redundancy, on the other hand it helps detecting information overlap between different application protocols and as such facilitates their integration. Depending on the application context, i.e. the collection of processes that is supported by the application protocol, an ARM can contain one or more units of functionality. All units of functionality are related to the processes defined in the AAM, that require each functionality. Figure 2 shows the relationship between data classes identified as information flows within the AAM and the UoFs derived from these data classes.

Figure 2 Relationship between information flows identified in the AAM and the structuring of the ARM into UoFs.

In the case of AP214, the ARM is divided into 35 UoFs. These UoFs represent modules that contain e.g. basic product management data, work management data, process plans, geometric models such as brep or surface models, surface conditions, geometric tolerances and different types of form features.

Once the overall structure of the ARM into UoFs has been fixed, the ARM is developed with the input of the application domain experts and based on their terminology detailing the information flows of the AAM identified to be in scope. The development process has to ensure integrity and self-consistency of the ARM that has to be an integrated conceptual model for the considered application domain. An overview of current methodologies for the development and integration of such models can be found in (Polly, 1996). To ensure consistency not only within one, but among multiple application protocols, the reuse of the common integrated resources by the mechanisms integration and interpretation have been introduced. The

fundamental principle of the data integration aspect is the common use of product data constructs provided by the Integrated Resources across the AIMs of all application protocols. During the interpretation process, the information requirements described in the ARM (application objects and assertions) are mapped to suitable constructs of the Integrated Resources, where necessary refinements and/or restrictions in form of new subtypes and constraints (local or global rules) are identified. It is part of the integration task to identify potential overlaps with other application protocols, that manifest themselves in the form of overlap in the AAMs, similar UoFs, and/or correspondences between the ARMs, and to ensure that same information requirements result in the same selection and specialization of integrated resource constructs. In case the overlaps result in new subtypes, AICs containing these new subtypes have to be defined and incorporated into the AIM.

The result of the interpretation process is documented in two components of the application protocol, the mapping table and the AIM. The mapping table provides the correspondences between the application specific information requirements described in the ARM and the constructs of the AIM that fulfil those information requirements and thus ensure traceability of the product information to be exchanged or shared on basis of the AIM back to the information requirements specified for the specific application domain in the ARM. A mapping table contains one table per UoF that is organized into five columns. The first column contains a separate row for all application elements, i.e. application objects (i.e. ARM entities and attributes) and assertions (i.e. relationships between entities) of the UoF, in alphabetical order of the entities. Attributes and assertions directly follow the entities for which they are defined. The second column contains the AIM element(s) to which the application element maps with the indication of the source schema in which the AIM element has been originally defined. Possible sources can be the integrated resource Parts, AICs or the application protocol itself. Column 4 of the mapping table contains numerical references to the global rules that constrain the valid instantiation of the given mapping in the AIM. The numerical references correspond to a numbered list of all global rules defined in the AIM.

Because of their different structure and terminology and the potential use of different modelling languages, there is often not a direct correspondence between ARM and AIM constructs. In case the particular ARM requirement is specified on a lower level of detail than the integrated resources, the mapping of an application element can require a combination of AIM elements (AND situation) or give alternative mappings (OR situation).

In addition, it might be necessary to express the particular application element by a related set of AIM constructs. How the AIM constructs have to be related to fulfil a particular information requirement is specified in column 5, the so-called reference path column. A reference path syntax has been developed to document the AIM structure necessary to fulfil the information requirement described by the particular application element.

In case an Application object maps to an AP specific subtype, the Reference path column contains the reference path from the subtype to its supertype from the Integrated Resources. In case the application element is a simple attribute that maps to an attribute of another AIM entity than the Application object maps to, the Reference Path column shows the reference path from the AIM element of the Application object to the AIM element of the simple attribute. In case the application element is an assertion, i.e. a relationship between to ARM entities that map to different AIM constructs, the AIM element column contains the word 'Path' and the reference path column documents the AIM constructs and their relationships necessary to establish the relationship on the AIM level, i.e. it contains the reference path starting at the AIM element the "From" Application object maps to and ending at the AIM element which the "To" Application

object maps to. Mapping rules can be used to specify that attribute values are restricted or certain subtypes are required only for a certain reference path.

The symbols used within a reference path definition, are shown below.

- [] : Multiple AIM elements or sections of the reference path are required to satisfy an Information requirement (AND situation).
- () : Multiple AIM elements or reference path sections are (exclusive) alternatives within the mapping to satisfy an information requirement (OR situation).
- { } : Enclosed section constrains the reference path to satisfy an information requirement (Mapping Rule).
- > : Attribute references the entity or select type given in the following row.
- <- : Entity or select type is referenced by the attribute in the following row.
- [i] : Attribute is aggregation of which a single member is given in the following row.
- => : Entity is a supertype of the entity given in the following row.
- <= : Entity is a subtype of the entity given in the following row.
- = : The string, select, or enumeration type is constrained to a choice or value.
- \ : Line continuation character.
- 1) : Footnotes are used to provide additional information.
- newline : Entity has attribute documented in the following row or attribute is attribute of entity documented in the following row.

A more detailed description of the mapping table documentation can be found in (ISO TC184, 1996-1) and (Anderl, 1995). As an example, a simplified extract of the ARM is shown in figure 3. The corresponding mapping table for this ARM extract is shown in figure 4.

ARM extract in EXPRESS-G:

Figure 3 Simplified extract from the mapping table of AP214.

Alongside the development of the mapping table, a so-called shortform of the AIM is developed. It contains references to the integrated resource constructs and to the AIC schemas to which a mapping exists (in form of the 'USE FROM' construct defined in EXPRESS), as well as the (textual and EXPRESS) definitions of AP specific types and entities and the global rules to restrict the valid sets of instances for the application domain. With the integrated resource and AIC schemas as input, this AIM shortform can be expanded to a self-contained longform according to the interface rules defined in (ISO 10303-11, 1994). This step might bring in entities, that are not specified in the AIM element or reference path column, but referenced by other entities.

Mapping table form_feature_with_explicit_shape UoF (FF3)

Applic. element	AIM element	Source	Rules	Reference path
HOLE_BOTTOM_CONDITION	hole_bottom	214		hole_bottom <= shape_aspect
bottom_type	shape_ aspect.name	41		hole_bottom <= shape_aspect shape_aspect.name {(shape_aspect.name = 'radiused') (shape_aspect.name = 'flat_with_radius') (shape_aspect.name = 'flat') (shape_aspect.name = 'pointed')}
ROUND_HOLE	round_hole	214		round_hole <= machining_feature <= shape_aspect
maximum_depth	measure_ representation_ item	45		round_hole <= machining_feature <= shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <= property_definition.definition property_definition <= property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_ representation.used_representation -> representation {representation => shape_representation} representation.items [i] -> representation_item => {representation_item.name = 'maximum_depth'} measure_representation_item {measure_representation_item <= measure_with_unit => length_measure_with_unit}
round_hole to hole_ bottom_condition (as hole_ bottom_condition)	PATH			round_hole <= machining_feature <= shape_aspect <= shape_aspect_relationship.relatng_shape_aspect shape_aspect_relationship {shape_aspect_relationship => feature_component_relationship} shape_aspect_relationship.related_shape_aspect-> shape_aspect => hole_bottom

Figure 4 Simplified extract from the mapping table of AP214.

Figure 5 shows an EXPRESS-G diagram of the AIM derived from the example given in figures 3 and 4. It can easily be seen, that the result of the mapping from the ARM to the integrated resources that are defined independently from a specific application context, is a more complex

structure that is more difficult to understand than the original structure specifically defined for the application context.

Figure 5 AIM subset derived from the figure 4 example (simple attributes not shown).

4 GENERATION OF AIM AND AIM SUBSETS FOR CONFORMANCE CLASSES

Although the mapping tables as currently documented in the application protocols are only described in a semiformal way, they allow for the automatic derivation of the information structure and part of the constraints of the AIM, e.g. which AIM constructs are existence dependent from other constructs. For this purpose, a tool has been developed, that is based on an own syntax format describing the mapping table. This tool has already been used to generate the EXPRESS shortform for the complete AIM of AP214 and also to create specific AIM subsets dedicated to the needs of particular computer applications used within the process chain supported by the AP. The subset generation is done on the basis of conformance class definitions which are based on the UoF structure of the ARM.

A conformance class is a definition of a subset of the AIM that may be used as the basis for implementation. In order to meet the needs of different computer applications used within a given application domain, two or more conformance classes may be defined for an application protocol, where if no conformance class is specified, a conforming implementation has to implement the complete AIM. As the definition of an implementation specific subset should be done from the application's point of view, that is described in the ARM, the AP214 approach is to specify the conformance classes on the ARM level, i.e. as combinations of UoFs. To derive the corresponding AIM subsets, that have to serve as the basis of the actual implementations,

the correct alternatives have to be extracted from the information contained in the mapping tables. For this purpose, the mapping table had to be extended by some additional information e.g. capturing the dependencies and/or correspondences between the implementation specific view, i.e. the conformance class definition based on the UoF structure, and the alternatives in the mapping documented as OR cases in the mapping table. Figure 6 shows the input and output for the AIM and conformance class generation.

Figure 6 Input and output for the generation of implementation specific conformance classes.

Although the current mapping table definition serves with some extensions as basis for the generation of the AIM subsets for the conformance classes, which provide the normative specification for the product data to be exchanged between computer applications that support the particular conformance class and also the conceptual schema for data sharing implementations, it still has some lacks e.g. the fact that the current mapping syntax is not specified in an unambiguous and formal way, that it cannot be exactly described how an instantiation of the AIM for a given ARM instantiation has to look like, and that a bidirectional mapping, i.e. from ARM to AIM and vice versa, cannot be specified.

In parallel to the current efforts to improve the mapping table syntax used, there have been ongoing efforts in and outside of the STEP arena to define mapping languages, that formally define how instances of a mapping target schema can be generated out of an instantiated source schema (unidirectional mapping) and eventually vice versa (bidirectional mapping). They are based on different approaches and therefore suited to perform in different areas. Examples of such mapping languages are EXPRESS-V (ISO TC184, 1995-3), EXPRESS-M (ISO TC184,

1995-1), and EXPRESS-X (ISO TC184, 1996-3). As an example, EXPRESS-V is mentioned here that seems to be suitable to compensate some lacks of the current mapping table syntax definition. A possible usage scenario for the integration of computer applications based on a single application protocol is described together with a proposal how the scenario can be realized using the information contained in the mapping tables.

5 GENERATION OF APPLICATION SPECIFIC VIEWS

EXPRESS-V has been developed to accommodate simplified views of a product model that omit unnecessary details of the model, that are not needed by particular computer applications. Using such a view is conceptually easier to understand and process within the application system. As the applications also need the possibility to update the data they access through their view, the mapping proposed is bidirectional.

Within AP214, the complete ARM represents an integrated schema to describe the information requirements of all applications supported by the AP on a conceptual level. An application specific view can be seen as one conformance class definition on the ARM level, i.e. as combination of those UoFs that are supported by this particular application. As described above, an implementation specific AIM subset corresponding to a set of conformance classes supporting a given set of applications to be integrated can be derived from the mapping table. It can be used as the integrated schema for a data sharing implementation for this set of applications. To generate the application specific views on the implemented schema, a mapping is necessary from the implementation back to the application requirements. Such a mapping can be specified using EXPRESS-V. The mapping table documented in the AP seems to be a good starting point for this task. Although it does not capture the complete instantiation relationships between ARM and AIM, it describes their semantic correspondences in an easy understandable way and ensures that an unidirectional mapping exists for all ARM constructs. Current work deals with extending the mapping table specification to capture more semantics to document the relationships between the original application specific requirements and their corresponding AIM constructs in a more precise way in order to derive a 'backwards' mapping from the AIM, i.e. the schema used for implementation, back to the application specific requirements specified in the ARM. Based on these extensions, EXPRESS-V schemata can at least be partially derived, each of them specifying the mapping between the AIM schema that acts as basis for the data sharing implementation, and a conformance class specified in the terminology and from the viewpoint of a specific application as combination of UoFs on the ARM level. From the completed EXPRESS-V specifications software can be derived to perform the mapping at runtime.

6 CONCLUSIONS

The integration of product life cycle views on the basis of an integrated product model specification becomes more and more important. Particularly for the support of product development processes, views dedicated to support the various types of applications used during the different life cycle phases are strongly required. The approach of integrating product life cycle views on the basis of a shared conceptual information model takes into account the data specification levels of ARM and AIM introduced by the current ISO 10303 architecture.

Although the mapping table notation as currently used to document the ARM to AIM mapping has several lacks, it is relatively easy to understand and it provides a systematic approach and structured way to document the mapping ensuring that for each construct of the source model, i.e. the ARM, a mapping specification exists to the target model, i.e. the AIM. This AIM does not exist at the beginning of the mapping process, but is created during this process 'on the fly'. Therefore, the mapping table is considered to be a suitable starting point for the mapping process, that has to be extended by further capabilities such as capturing instantiation correspondences. The extended mapping table can then be used to generate (at least partially) mapping schemas in a formal mapping language like EXPRESS-V or EXPRESS-M, depending on the purpose of the implementation to be supported. Out of this formal specification, software can be derived to perform the mapping. Current work in this area include the specification of extensions to the mapping table syntax in this direction and the development of tools that derive such formal specification based on the mapping table notation.

7 REFERENCES

- Anderl, R. and Wasmer, A. (1995) Methoden zur Modellintegration. *Informationstechnik und Technische Informatik 5/95*, Oldenbourg Verlag, Munich.
- ISO 10303-1 (1994) Part 1: Overview and fundamental principles.
- ISO 10303-11 (1994) EXPRESS Language Reference Manual.
- ISO 10303-202 (1996) Part 202: Application protocol: Associative Draughting.
- ISO TC184 (1995-1) EXPRESS-M Reference Manual. ISO TC184/SC4/WG5/N243.
- ISO TC184 (1995-2) Part 214: Application protocol: Core Data for Automotive Mechanical Design Processes. ISO TC184/SC4/N319.
- ISO TC184 (1995-3) EXPRESS-V Reference Manual. ISO TC184/SC4/WG5/N251.
- ISO TC184 (1996-1) Guidelines for the development of mapping tables. ISO TC184/SC4/N436.
- ISO TC184 (1996-2) Description methods: *architecture and development methodology reference manual*. ISO TC184/SC4/WG10/N62.
- ISO TC184 (1996-3) EXPRESS-X Reference manual. ISO TC184/SC4/WG11/N002.
- Polly, A. (1996) Methodische Entwicklung und Integration von Produktmodellen. Shaker Verlag, Aachen.

8 BIOGRAPHY

Prof. Dr.-Ing. Reiner Anderl received his PhD in 1984 and was working as chief engineer at the Institut für Rechneranwendung in Planung und Konstruktion (RPK) at the University of Karlsruhe. In 1991 he got the appointment to a university lectureship in the area of CAD/CAM and product data technology. Since April 1993 he is professor for computer integrated design at the 'Fachgebiet Datenverarbeitung in der Konstruktion DiK' at the University of Darmstadt.

Anne Wasmer received her degree in computer science at the Technical University of Karlsruhe where she worked as research assistant at the 'Institut für Rechneranwendung in Planung und Konstruktion' (RPK) since 1992. Since April 1993, she has been working as a research assistant at the 'Fachgebiet Datenverarbeitung in der Konstruktion'.