Journal: Systems Engineering, Vol. 8, No. 1, 2005
Wiley Periodicals, pp. 41-50

*Regular Paper*

# Model Transfer Among CASE Tools in Systems Engineering

**Roland Eckert,[1,*] Wolfgang Mansel,[2] and Günther Specht[3]**

[1]EADS Deutschland GmbH, MT332, 81663 Munich, Germany

[2]EADS Deutschland GmbH, MT33, 81663 Munich, Germany

[3]University of Ulm, Department of Databases and Informationsystems, 89069 Ulm, Germany

Received 23 April 2004; Accepted 24 September 2004, after one or more revisions
Published online in Wiley InterScience (www.interscience.wiley.com).
DOI 10.1002/sys.20018

## ABSTRACT

Systems engineering as a core discipline of technical product development uses a variety of different Computer-Aided Systems Engineering (CASE) tools. Very often the need arises for data transfer between such tools. Currently, due to insufficient interface formats, a labor-intensive manual transfer of data is necessary. A step towards improving data exchange between systems engineering tools along the process life cycle is the definition of standard interface formats such as the application protocol (AP) framework of ISO Standards for the Exchange of Product Data (STEP). The aim of this paper is to analyze and evaluate the implementations of the Systems Engineering Data Representation and Exchange Standard (SEDRES/AP233) information model with regard to supporting data exchange, to propose improvements for the emerging modular AP233 ISO standard and to provide solutions for the observed problems. The solutions proposed in this paper were developed while designing, developing, and using a STEP ISO 10303–SEDRES conformant import and export interface for the tools Teamwork, Statemate, and the requirement management tool DOORS. The developed interfaces demonstrate that the SEDRES information model is an efficient medium for tool and vendor independent transfer of systems engineering information. © 2004 Wiley Periodicals, Inc. Syst Eng 8: 41–50, 2005

Key words: STEP; SEDRES; Computer-Aided Design; interoperability; data translation; standards

*Author to whom correspondence should be addressed (e-mail: Roland.Eckert@EADS.com).

## 1. INTRODUCTION

For the success of most complex engineering projects, a close cooperation of all involved parties by data exchange is crucial. Often, on large projects, a multi-tool environment is used or a tool environment change is required during the project life cycle. Often an automated data exchange is not possible, because the semantics of the source and the target data schema are too different or the data schemas are proprietary. Then only manual data transfer is possible, but manual transfer of data increases the likelihood of errors and wastes time. It can take so long that users may decide it is not worthwhile to propagate every change. Instead, they may work with out-of-date, or incomplete data. Therefore, within the Systems Engineering discipline across the majority of industry sectors, there is an increased requirement to improve data transfer across company, international and environment boundaries.

STEP, ISO 10303 [ISO, 1994, Kemmerer, 1999] is a comprehensive series of documents, which provide industry with solutions to exchange and share the information used to define a product throughout the entire life cycle of the product. The STEP standard defines an integrated information model, which supports multiple views of product data for different applications. For each application area covered by the standard, a standardized Application Protocol (Part 201-238) defines the information required for a specific application. The implementation methods for data exchange are also defined, e.g., by ISO STEP 10303—Part 28 (XML), Part 21 (ASCII) or SDAI.

Earlier systems engineering standards such as [IEEE 1220, 1999] and [ANSI/EIA 632, 1998] are primarily focused upon the development aspects of technical systems. [ISO/IEC 15288, 2002] is focused on the definition of a set of processes that can be tailored to describe the actions of organizations and individuals at any point throughout the systems life cycle. However, a standard for the design data exchange for the systems engineering domain is missing. This led to the proposal of the STEP Systems Engineering Data Representation and Exchange (SEDRES) project funded by the Information Society Technology (IST) Framework IV of the European Commission [SEDRES, 1999]. It resulted in a proposal for improved data exchange and interoperability of data between different computer-aided systems and software engineering tools. SEDRES/AP233 is in the process of becoming a Publicly Available Specification (PAS) under ISO/AP233. A modular-AP233 is currently under development in the framework of ISO/SC4/WG3 and is being released sequentially in a series of module sets. The SEDRES model is an important predecessor to the AP233 standard and its implementations are sufficiently robust for use on projects and for demonstration of economically useful results.

The industrial motivation for supporting SEDRES and standardizing and developing AP233 interfaces (gateways) is that we envisage an environment facilitating the sharing of systems engineering information between relevant stakeholders, regardless of the (current) barriers arising from the lack of open semantic standards and lack of interoperability of tools and PDM environments. The individual systems engineering activities are frequently supported by different tools and tool environments, for instance, tools for requirements analysis, or for functional analysis. Also, different tools for the same activity are used by different partners, or even by the same partners in different stages of product development (for instance, for functional modelling during concept development, versus full product definition) due to different needs for design fidelity or analysis. However, the semantic-level integration between these tools is poor or nonexistent. AP233 aims to be the solution to build up an engineering database for an integrated development process.

The life cycle of a project, from system requirements until retirement, disposal, and replacement, is often much longer than the life span of tools or tool versions. Because of the product liability, it is often necessary to have the design models available even after the retirement of such a product. A tool-neutral storage format of the model is a solution to this archiving problem.

A large number of papers [e.g., Johnson, 2003a, 2003b; Herzog, 2004; Eckert and Mansel, 2004; Eckert and Specht, 2004] describe the potential benefit of the use of AP233 for data exchange in the systems engineering process. However, these papers include only a few concrete examples of data exchange examples within one tool family (e.g., requirement management tools). There are no published reports about data exchange results, the quality of the exchanged data, or solutions for improving the results.

This paper reports the experiences of SEDRES-based data exchange between the tools Statemate, Teamwork, and DOORS, which are typically used in the application field of electronic and software systems. The reader gets an impression, why it is not a trivial job to develop an interface in the systems engineering domain, and how SEDRES is being improved and has to be improved further for the developing modular AP233.

The rest of the paper is organized as follows: Section 2 introduces the SEDRES information model. Section 3 describes SEDRES Application experiences. Section 4 discusses the SEDRES Information Model and its implementation. Section 5 closes with a summary and a conclusion.

## 2. MODEL TRANSFER PROBLEMS AND THE SEDRES INFORMATION MODEL

The general problem of product data exchange between tools is that data can get lost, can become interpreted incorrectly, or can be displayed incomprehensibly during automatic translation. If the user loses 1% of the information without knowing which part, he has to check the whole model. This additional time expenditure dramatically reduces the benefit of an automated data exchange. The data loss is a result of incorrect semantic mapping, technical tool incompatibility, and inability to display the received information in user-readable form. These reasons reduce the value of the received data significantly. Subtle and unnoticed transmission errors can lead to expensive errors in the systems development. Therefore, the data exchange between several tools evokes a number of challenges. These challenges can be divided into problems caused by differences in the semantics of the data models, into heterogeneous schemas and interoperability.

*Semantic interoperability* presents a collection of issues, all of which become more pronounced as individual resources—each internally constructed in their own semantically consistent fashion—have to be made available through "gateways" such as the application protocols from STEP.

*Technical Interoperability*, in many ways the most straightforward aspect of maintaining interoperability, incorporates communication, transport, storage, and representation. This is the intended scope of standards such as Z39.50 [Z39.50, 1992], Interlibrary Loan (ILL) Protocol (ISO 10160/10161) [Jackson, 1997], and ISO 10303-28 (XML). A more detailed analysis of the interoperability challenge can be found in Brownsword [2004] and Morris [2004].

For overcoming the technical interoperability, SEDRES can be used as a semantic description of the data that uses, for example, ISO 10303-21 (ASCII) and ISO 10303-28 (XML) as a way of implementation. Only the application protocols provide the definition of information relevant to an engineering domain, since they define semantics for an industrial domain rather than being syntax-oriented or application-generic like XML, SDAI, etc.

The main intention of SEDRES is to overcome the semantic interoperability. The data model of AP233 supports, among others, the following "concepts":

- System and subsystem views including hierarchies
- Requirements: text and model-based and requirements traceability
- System behavior and functional architecture
- Functional decomposition, interfaces, and allocation
- Functional and physical flows; behavior models; finite state machines
- System architecture and interface control
- Component decomposition, interfaces, and allocation
- Parts libraries & product lines

If these concepts are not adequate, they can be extended by concepts from other application protocols, like, e.g., Product Life Cycle Support ("AP239"). (In future, the available application protocols will be modularized and interchangeable.)

Data exchange between heterogeneous tools becomes a challenge due to the different semantics of the data models of the tools. This is the consequence of different concepts and database representations as a result of their independent development by vendors. We identify semantic heterogeneity, the lack of semantic interoperability, and the lack of technical interoperability as such challenges. We identify two types of semantic heterogeneity.

*Cognitive heterogeneity* arises when two data models have different perceptions of real world facts. Using the same names, i.e., homonyms, can conceal these differences. For example, the field name "balance" may represent the checking balance in the checking accounts application. The same field name "balance" may also represent the savings balance in the savings accounts application. In this case, the term "balance" is a homonym.

*Naming heterogeneity* refers to different names for identical concepts of real world facts, also called synonyms. As an example, consider how a student may be referred to in a college data system. In the student registration system, the student may be referred to by the field name "student-number." On the other hand, in the majors and graduation system, the student may be referred to by the field name "candidate-number." The terms "student-number" and "candidate-number" are synonyms.

SEDRES offers a solution for the cognitive heterogeneity. The granularity of SEDRES is so high that it was possible to find separate units of functions for homonyms. SEDRES also offers solutions for naming heterogeneity, called synonyms. The real world facts are clearly defined in the documentation of the SEDRES publicly available specification PAS. Every interface developer and CASE tool provider can access this information and adapt his data model. The SEDRES data model delivers unambiguous definitions.

The intention of SEDRES is to facilitate faster and cheaper development of translators. SEDRES acts not

only as a simple exchange standard. It also captures the semantics of systems engineering information and supports exchange of overlapping data between different classes of tools (i.e., tools for system requirements in text and tools for modeling requirements and design). It facilitates traceability and management of systems engineering information across different tools. It opens up the possibility of creating meaningful central data repositories. It minimizes cost of data exchange, data re-entry, and errors. It improves the quality of systems engineering information exchange. SEDRES includes product and process data to support the enerprise and is easily partitioned.

It is only possible to map information (e.g., entities) from the source to the target tool where an equal or suitable structure, at least for a subset of the stored data, is available. *Data loss* during a data transformation occurs when information (e.g. entities) are mapped from the source to the target tool, where no equal or suitable structure is available. Examples are given in the following schema of mapping classes:

**Legend:**

$\varnothing$ empty set
$\in$ belongs to
$\neg$ logical **not**
$\exists$ logical for some (there exists)
$\forall$ logical for all
$\rightarrow$ logical imply

$$b = T(a)$$

with

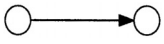
$$a = \{a_1, a_2, ..., a_m\} \in \textit{Schema A} \vee \varnothing \wedge$$

$$b = \{b_1, b_2, ..., b_n\} \in \textit{Schema B} \vee \varnothing$$

If a pair of schemas (A, B) is considered, then a transformation function $T$ can be defined to capture how a specific concept in schema A shall be represented in schema B.

A schema may be defined with the intention of implementing it in a particular database system, in which case it is called a data model. An information model or concept model is a schema that is independent of any particular implementation.

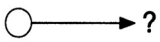
**Equality**



A class for the mapping functions whose application results in equivalent semantics in the source and target schema.

$$\forall a \in A : \exists\, b \in B : a \rightarrow b.$$

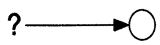
**A is more expressive than B**



A class for the mapping function for which at least one source element exists with no representation in the target schema.

$$\exists\, a \in A : \neg \exists\, b \in B : a \rightarrow b.$$

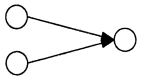
**A is less expressive than B**



A class for the mapping function for which there exists, for some or all target elements, no representation in the source schema.

$$\exists\, b \in B : \neg \exists\, a \in A : a \rightarrow b.$$

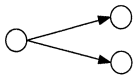
**Aggregator**



A class for the mapping function for which some or all source elements are merged into one representation in the target schema.

$$\exists\ a_1 \in A, \exists\, a_2 \in A, a_1 \neq a_2 : \exists\, b \in B : (a_1 \rightarrow b) \wedge (a_2 \rightarrow b).$$

**Dispatcher**



A class for the mapping function for which a single element of the source schema is split into two or more representations in the target schema.

$$\exists\ b_1 \in B, \exists\, B_2 \in B, B_1 \neq B_2 : \exists\, a \in A : (a \rightarrow b_1) \wedge (a \rightarrow b_2).$$

## 3. SEDRES APPLICATION EXPERIENCE

EADS Deutschland GmbH has validated the SEDRES interfaces for the system and software design tools Statemate, Teamwork, the requirement management tool DOORS, and the proprietary PDM tool Teamcenter [Eckert and Johansson, 2003; Eckert and Mansel, 2004; Eckert and Specht, 2004], which are typically used in an engineering environment. The scenario is that different companies use Statemate and Teamwork in different life-cycle stages. DOORS is a common requirement management tool.

The application context for the validation scenario was an aircraft landing gear control specification. This provides a relatively rich functional and behavioral specification, which was exchanged between the tools.

Table I shows the effort required to develop an import and export interface between the tools using

**Table I. Effort for Developing SEDRES-Tool Import and Export Interfaces**

| Tool: | Teamwork | Statemate | PDM | StP | DOORS |
|---|---|---|---|---|---|
| Tool Data Model | proprietary | ASCII Files | EXPRESS | proprietary | proprietary |
| Mapping Tool | from Combitech | EDM | PDMConnect | PROSTEP Suit | ECCO |
| Effort: [man-month] | 7.5 | 6 | 4 hours! | 6 | 5 |

SEDRES/AP233. The effort for developing a universal interface between several requirements management tools and the AP233 modules for the textual-based requirements and the property based requirements module set [Johnson, 2003a], has been about 1 man-week. This low cost is the result of building and supplying an Interface Development Tool to interested vendors [Eurostep, 2004].

Four different data sets were tested and are described in the Table II. The first column describes the data export from Statemate (St) and the second column describes the data export from Teamwork (Tw), after the data from Statemate was imported into Teamwork. These metrics describe fundamental elements of the used design data. After the data exchange between different tools via AP233, these metrics are changed. The "Function Instances" are nodes in a hierarchical structure. The "Leaf Functions" are nodes that are no longer subdivided and have no further child functions. "Defined Functions" are data flows between the functional nodes. The "Element Identifier" identifies elements under configuration control. The critical observation is that "Defined Functions" get lost. Both tools use different concepts for "Element Identifier." The different number of instances is a result of the different concepts of the gateways and the way data is stored in the tools.

**Table II. Design Data Exchange Between Statemate and Teamwork via AP233**

| Data | Set1 | | Set 2 | | Set 3 | | Set 4 | |
|---|---|---|---|---|---|---|---|---|
| Tool export | St → | Tw→ | St → | Tw→ | St → | Tw→ | St → | Tw→ |
| Instances | 3374 | 5537 | 5527 | 8323 | 9033 | 14566 | 5376 | 8900 |
| Function Instances | 36 | 36 | 61 | 61 | 114 | 114 | 62 | 62 |
| Leaf Functions | 30 | 30 | 51 | 51 | 86 | 86 | 48 | 48 |
| Defined Functions | 188 | 161 | 351 | 319 | 670 | 667 | 347 | 347 |
| Element Identifier | 431 | 740 | 652 | 1051 | 957 | 1736 | 572 | 1113 |

The results of the validation scenario provided important insight both into the strengths and weaknesses of the data model and into several of the appropriate technologies relevant to shared data environments. Important observations and results were obtained, which can be organised into implementer perspectives and engineer perspectives.

Primarily from an implementer perspective:

a. In all cases, the data model was found to be implementable, the implementers being able to map between tool concepts and corresponding concepts within the data model. As such the practical prototyping is a general validation of the data model from an implementation perspective (Table I).
b. It was possible to produce a basic SEDRES-based information server. This has revealed the need for further work in the areas of item identification and product data management.
c. Systems engineering information from the validation scenario (Table II) was successfully exchanged between tools (Table I). In the case of the functional information, not only the functions and their definitions were exchanged, but also the flows and the graphic representation information. This shows the practical capability of the data model, the STEP technology of flat-file exchange (to be compared also with the data browsing capability), and the feasibility of practical tool interfaces.
d. As SEDRES-enabled design data exchange becomes a reality, it reveals the increasing chal-

lenge of configuration control. The current data model reveals the limitations to which current tools support comprehensive fine-grained data management across sets of tools.

e. On the process side, the validation scenario shows how traceability between data from different design tools can be built into the design and then proven through crosschecks. This also illustrates the possible use of the data model: the traceability is captured by the design tool (the source) and then recomputed by the traceability tool, hence supporting the verification process.
f. In order to fully exploit the standards potential use, SEDRES can be used to link data from Statemate, Teamwork and DOORS to a process management tool such as "Teamcenter" [Eckert and Mansel, 2004; Eckert and Specht, 2004]. Such a tool would be responsible for managing the design process, the design tools providing design support and traceability tools providing the verification facility.

As a result of the validation scenario we have found that the current AP233 data model is restricted in several domains and has to be improved:

- Graphical representation
- No unique item identifier across a project
- No concept of object order
- Does not know from which CASE tool data was delivered

*Graphical Representation:* The graphical representation of the design objects in the tools Statemate and Teamwork differs. In Statemate the basic symbols are rectangles, while in Teamwork basic symbols are circles. Because of this, the proportion of shapes, text, workspace, etc. differs. The tools use different predefined symbols that are unknown in other tools.

One feature of the interface is the possibility to choose between the use of the graphical representation of the original model or auto-generating a new Teamwork-specific graphical representation. With the generation of the Teamwork-specific graphical representation it is possible to compensate the mismatch of graphical representation between different tools. Using the interface-generated graphical representation, the orientation of symbols is optimized for the presentation in Teamwork. On the other hand, the symbols are not located in the same order on the screen as in the originating tool. However, the hierarchy of the design objects (symbols) is not affected.

One problem of going from Teamwork to Statemate is that in Teamwork external data can start anywhere in the empty plane, whereas in Statemate external data requires external activities to define a start. At present there is no automatic feature for generating a clearly arranged graphical representation. After an import, engineers have to make a time and cost-intensive rearrangement of the diagrams.

Statemate can represent concurrent states as a single "and state." On transfer to Teamwork Real Time, this "and state" must be represented by a factorial expression into individual states. Layout and display of the differences in the two tools is problematic.

*No unique item identifier across a project:* The "Identity" had no focus in SEDRES. For example, in a neutral systems engineering repository, merging the data from different tools requires a unique identifier. The use of the PDM Module could compensate for these deficiencies, but this has to be evaluated. This, however, will be problematical, because it will be difficult for the target tools to interpret this identifier). A solution for this could be a data dictionary [ISO/IEC 11179-6, 1997] that maps the identifier for each individual CASE tool.

The way in which identification and versioning is handled, and the mechanism for building hierarchies (functional) in SEDRES [ISO, 2000] is unusual in the STEP domain. The STEP/PDM Schema, and the reusable small STEP EXPRESS modules are in use in the AP233 standard under development.

*No concept of object order:* It is not possible to reliably maintain the order in which requirements and other items are presented. In SEDRES the only ordering of functions occurs in the "Functional Behavior" part of the model. The current AP233 development has released module sets for Test Based Requirements and Property Based Requirements that enable reconstruction of the document structure when data is exchanged. These modules are balloted and released with an available demo tool and a tool for economic interface development. Seven requirements management tools have existing interfaces to these AP233 module sets.

*Does not know from which CASE tool data were delivered:* There are mismatches between tools and the SEDRES data model where the capabilities of Systems Engineering design tools differ (poorer or richer) to those captured in the SEDRES data model. Most significantly, many Systems Engineering tools do not support the concept of alternative solutions to a problem in a manageable way.

*Further general observations* came from validation scenarios where the tools Teamwork and Statement were involved. Teamwork, in contrast to Statemate, has no concept of ports. Therefore, it was necessary to synthesise ports for mapped flows and bubbles to create valid SEDRES data. SEDRES needs a port decomposition capability along with a need to get a fine grain view

of port-to-port connectivity at each associated level of abstraction. A Port is a connection point on a system assembly in the system assembly decomposition hierarchy. The AP233 module set for Structure (under development) supports these needs along with allocation of interface requirements to the port connections and the association of emergent properties at the port interconnection.

The Teamwork export interface from Conformics is synthesizing data instances because a data instance is not a concept in the Teamwork model. For each synthesized data instance the Teamwork synthesizes a new element identifier, but it may reference an existing definition if an equal definition is available. Many flows will be broken if one interface is synthesizing data instances and others are not.

Teamwork does not have any concept of constant values. All values have to be stored globally. The export interface is always assigning the false value to this attribute. An important experience from the interface development was that all SEDRES interfaces have to support IO_COMPOSITION_PORT (Fig. 1). Otherwise, flows will get lost in some tools. Example: "a" is a flow that ends in a bubble "C". The flow is identified as flow "a" to the children (a.i, a.j, a.k) within bubble 'C'. The flows (a.i, a.j, a.k) will be broken if some interfaces are using the entity IO_COMPOSITION_PORT and some are not. For the described complex flows the SEDRES entity IO_COMPOSITION_PORT must be used.

The data structures of the tools Statemate and Teamwork differ. In Statemate the data is organized in hierarchical structures. In Teamwork the data are structured in a relational schema. There are many mappings between these structures that are syntactically correct and result in the transfer of the data from one structure to another. However, many of these are semantically incorrect, and cannot be fixed by AP233 and result in information loss. For example, a student can visit zero or more lessons. In the relational schema this is syntactically represented by foreign key linkage between the relations Student and Lesson on the ID attribute. In the hierarchical schema, it is represented by the fact that a Student record is the parent of a given Lesson record. Here, the ID attribute in Lesson serves only to sequence a list of lessons for each student. The syntactical translation from the hierarchical schema to the relational one maps each Student record to an equivalent one in the relation Student and each Lesson record to an equivalent one in the relation Lesson. This is semantically incorrect. ID in Lesson is not a valid foreign key to reference equivalents in the relation Student.

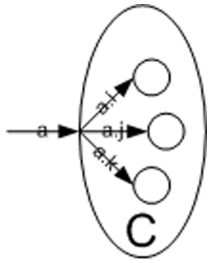



**Figure 1.** Port concept in Statemate.

After importing the data sets as AP233 conform P21-files (Table III), we have exported them again. The exported file was compared with the imported data and the result was described in Table III.

## 4. DISCUSSION OF THE SEDRES INFORMATION MODEL AND ITS IMPLEMENTATION

The complexity of the tool data model is a great obstacle for an interface designer. Independent of implementa-

**Table III. Data Exchange Results Described as Set Theory**

| from \ to | Teamwork | Statemate | PDM | StP | DOORS |
|---|---|---|---|---|---|
| **Teamwork** | equality | A<B | A>B | equality | A>B |
| **Statemate** | A>B | equality | A>B | A>B | A>B |
| **PDM** | A<B | A<B | equality | A<B | A<B |
| **StP** | equality | A<B | A>B | equality | A>B |
| **DOORS** | A<B | A<B | A>B | A<B | equality |

tion method, it must be carefully analyzed before an interface is capable of valid, high quality data transfer. In SEDRES the complexity is sorted out because it can be used as a well-documented reference data model and, as such, provides useful guidance for interface development. It is very difficult to perform a mapping from SEDRES to tool data models if the CASE tool provider hides the proprietary data models. It is also recommended to reconsider the established mapping during the implementation as new experience is gained. For an effective interface development it is necessary to know the tool mechanisms, the semantics of the tool and the supported processes.

The effort for developing a tool-AP233 interface (Table I) can be significantly reduced with familiarity of the involved data models before starting the development. Once the data model of AP233 is understood, it is a trivial job to develop new tool-AP233 interfaces. For the tool vendors it would be a benefit to support AP233 because of the significantly reduced costs for developing such tool interfaces.

The graphical representation capability of SEDRES data model [ISO, 2000] has to be improved. The absolute position information of model objects has to be replaced with relative position information. This is a topic of further development, e.g., in the course of the current AP233 development activities in ISO/SC4/WG3. (Another problem could be that the interfaces are not mature enough for the graphical representation topic.) If we look at tools for designing printed circuit boards, they use a sophisticated algorithm for improving the graphical representation. The results of this algorithm are much better than a human being can attain. The third, but unrealistic solution, is that all tools support the same graphical representation, as suggested in the UML 2.0 or Systems Modelling Language (SysML, www.sysml.org) initiative. Then all tools support an agreed set of symbols and it is no longer necessary to define mappings between different symbols like circles (Teamwork) and rectangular (Statemate). SysML is being developed in conjunction with AP-233 with the goal that SysML graphical models and the associated information in the model repository can be exchanged via AP-233 standard. Version 1.0 of SysML is expected to be adopted by the OMG with tool implementations to be introduced beginning in 2005.

The opinion of the authors is that the graphical representation is the most critical point. A model with a layout similar to that defined by a source tool is much more readable than any model layout synthesised by a low-level algorithm. This is a critical issue because systems engineers today use graphical representations of a system to communicate its functional and data requirements.

At present, no available tool can perform all the tasks needed for a full product life cycle. A close cooperation between different tools from different tool providers is vital. Because of the duration of development and time in service, no tool supplier is able to provide support over the years. So the openness and supported interface standards will be a purchasing factor for the industry. Large projects are usually conservative in changing the adopted tools and retraining teams of engineers because of the cost and risk involved.

We expect, in general, that the tool vendors will implement the interfaces and not the industry using the tool. For industry the interoperability of tools and worthiness and trustfulness of the export results will be a buying argument. Tool suppliers should take an active role in the development of the emerging AP233 to enrich the standard with their own extensive experiences.

In a sequential file-based data exchange between different tools, data loss cannot be precluded. It is not the fault of the neutral data model; it is because of the different capabilities of the design tools. This dramatically reduces the benefits of an automated interface. The data export from one tool to another tool is processed in several steps. At every step errors can occur that should be documented, e.g., semantic errors, syntax errors, broken business rules. A solution could be a standardised and machine-interpretable transformation or mapping report [Eckert and Mansel, 2004]. The benefits for such a report are:

- Visibility to the end user of the effects of the data exchange
- Worthiness and confidence of the delivered data
- Documentation of data loss
- List of delivered data
- Minimized error propagation
- History of data
- Constraint violations
- Handshaking function for data integration

The transformation report traces data exchange actions and is a precondition for data integration strategy to improve quality of exchanged information. The described report is not tool-specific, and therefore it is suggested that it be a part of the framework of ISO standards, e.g., “Standard for the Exchange of product model data” (STEP ISO 10303). The structure of the transformation report is generic, so it is easy to use the report without any adaptation for all kinds of data transformations.

The transformation report is not fixed by a special encoding, but is suitable to use a standard encoding like, e.g., XML (ISO 10303-28) or ASCII (ISO 10303-21).

Subsequent data exchange between the tools has limitations. One further exploitation, which remains to be investigated, is the use of AP233 as a neutral systems engineering repository [Eckert, 2002]. Application Protocols are also suitable for storing data in a neutral format. The tools could query the database to retrieve relevant information instead of the whole data of the source tool. If it is possible to merge the information from different tools into such a repository, new methods of consistency checks, syntactical correctness, and completeness checks of the designed model are possible. An open neutral repository based on AP233 offers a solution for the long-term data storage challenge.

## 5. SUMMARY AND CONCLUSION

The entire software community is well aware that the problems of interoperability are many and are not likely to disappear quickly. AP233, following the pioneering efforts of the SEDRES contracts, is one more step in the direction of overcoming the difficulties.

SEDRES provides a solid base for Systems Engineering data exchange and interface implementation. The interfaces and information transfers demonstrated with SEDRES that the emerging AP233 standard can be an efficient medium for tool and vendor independent transfer of systems engineering information provided that the defects reported here are addressed in the emerging AP233 standard. Many practical research results are available from the current SEDRES model that should be used in the development of the emerging AP233. If the defects are resolved in the AP233 standard, a new century of Systems Engineering will begin.

## ACKNOWLEDGMENTS

The authors thank all members of the SEDRES-2 project for their inspired work and for making possible the evaluation of the following interfaces:

- W. Scott, University South Australia, AP233—DOORS Import Interface, developed during SEDM (System Engineering design Methodologies).
- M. Giblin, BAE SYSTEMS, AP233—Statemate Interface, developed during SEDRES-2 project.
- Dr. G. Johansson, Conformics AB, AP233—Teamwork Interface, developed during SEDRES-2 project for EADS Deutschland GmbH.

## REFERENCES

ANSI/EIA 632, Processes for engineering a system, EIA, Arlington, VA, 1998.

ANSI/NISO Z39.50-1992 (version 2), Information Retrieval Service and Protocol: American National Standard, Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection, International Standardization Organization, Geneva, Switzerland, 1992.

L.L. Brownsword.; D. Carney, D. Fisher; G. Lewis, C. Meyers, E. Morris, P. Place, J. Smith, and L. Wrage, Current perspectives on interoperability, CMU/SEI-2004-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 2004.

R. Eckert, Call of EUC EOI.FP6.2002, NeSED, http://eoi.cordis.lu/dsp_details.cfm?ID=36203, June 6, 2002.

R. Eckert and G. Johansson, Experiences from the use and development of ISO 10303-AP 233 interfaces in the systems engineering domain, 9, Int Conf Concurrent Enterprises, Espo, Finland, 2003, pp. 501–508.

R. Eckert and W. Mansel, Integrating system engineering into PDM by AP233, International Council on Systems Engineering, Toulouse, France, 2004.

R. Eckert and G. Specht, Transformation Report: The missing Standard for Data Exchange, 6th NASA-ESA Workshop on Product Data Exchange, Friedrichshafen, Germany, 2004.

Eurostep, http://ap233.eurostep.com/software.htm, June 5, 2004.

E. Herzog, An approach to systems engineering tool data representation and exchange, Linköping Studies in Science and Technology, Dissertation No. 867, Linköping, Sweden, 2004.

IEEE 1220, IEEE standard for application and management of the systems engineering process, IEEE, New York, 1999.

ISO, ISO 10303-1, Industrial automation systems and integration—Product Data Representation and Exchange—Part 1: Overview and fundamental principles, International Standardization Organization, Geneva, Switzerland, 1994.

ISO, ISO WD5.1 10303-233, Industrial automation systems and integration: Product data representation and exchange: System engineering and design, International Standardization Organization, Geneva, Switzerland, 2000.

ISO/IEC 11179-6, Informationstechnik—Festlegung und Normung von Datenelementen—Teil 6: Registrierung von Datenelementen, International Standardization Organization/International Electrotechnical Commission, Geneva, Switzerland, 1997.

ISO/IEC 15288, Systems engineering—system life cycle processes, International Standardization Organization/International Electrotechnical Commission, Geneva, Switzerland, 2002, 7 pages.

M. Jackson, The application of the ILL Protocol to existing ILL systems, 63rd IFLA Gen Conf, Copenhagen, Denmark, September 2, 1997.

J.F.E. Johnson, The future Systems Engineering Data Exchange standard AP-233; Sharing the results of the SEDRES Project, INCOSE, 1999.

J.F.E. Johnson, How does AP233 support a systems engineering process (e.g. EIA-632)?, BAE SYSTEMS, INCOSE 2003, Washington, DC, 2003a.

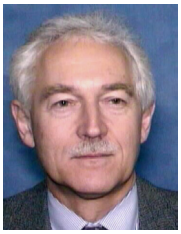
J.F.E. Johnson, AP233 requirements data exchange—lessons learned, Presentation, ISO SC4/WG3, 22,October 2003b.

S. Kemmerer (Editor), STEP—The Grand Experience, NIST Special Publication 939, Gaithersburg, MD, 1999.

E. Morris, L. Levine, C. Meyers, P. Place, and D. Plakosh, System of Systems Interoperability (SOSI): Final report, CMU/SEI-2004-TR-004, ESC-TR-2004-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2004.

SEDRES, Systems Engineering Data Representation and Exchange Standardisation-2, ESPRIT 20496 and IST-1999-11953; ISO 10303 AP233, Brussels, Belgium, 1999.

Mr. Roland Eckert received his master degree in computer science at the Technical University Ilmenau, Germany. He started his career at EADS in 1999 as software engineer in an avionics software development project. Further he worked in the EADS internal product data management project, contributed as project member to SEDRES-2 and is currently involved in a four-nation industrial process improvement project. He is German DIN representative for ISO activities.



Dr. Wolfgang Mansel obtained his master's degree and Ph.D. in physics at the Technical University of München. After several years in physics research, in 1985 he joined the aerospace industry and has been involved since then in the management of avionics software development projects as well as software technology projects with respect to software engineering, process improvement projects, and advanced information technology and product data management. He was the Project Manager of SEDRES-2 and now manages the enhanced process & toolset project. He holds the position of Software Technology Chief Engineer.

Professor Dr. Günther Specht is professor at the University of Ulm in the Department of Databases and Information Systems. He earned his Ph.D. in 1992 and his habilitation (Dr. habil.) in 1998, both at TU München. He held his first professorship at the TU München, then joined TU Ilmenau, and for 3 years has been at the University of Ulm. Research stays led him to TU Helsinki, Finland and IBM Almaden, California. His main interests include multimedia databases, mobile information systems, XML, ontologies, and systems engineering.