How does AP233 support a Systems Engineering process (e.g. ANSI/EIA-632)? An update

Dr Julian Johnson
BAE SYSTEMS
Systems Engineering Innovation Centre (SEIC),
Holywell Park, Loughborough, LE11 3TU, UK
julian.johnson@baesystems.com

Abstract. The development of the Systems Engineering data standard AP233 was initiated and pursued by the work of the SEDRES and SEDRES-2 projects during 1996 through 2001. Subsequently Modular-AP233 has been under development, building on the experience of the SEDRES/AP233 data model, and extending the scope with additional Systems Engineering concepts. This document explains how a data standard such as AP233 provides support to a practical systems engineering process, in at least two ways, firstly by providing clarity to systems engineering concepts and secondly by enabling practical support to interoperability of systems engineering support tools. The support is illustrated primarily in the context of ANSI/EIA 632. The original paper is brought up to date with the current status, challenges, and personal observations.

Note

The original 2003 paper was co-authored with Dr Ian Bailey, then of Eurostep Ltd., now of Model Futures Ltd, ian@modelfutures.com.

Introduction

Systems engineering (SE) developments involve the creation of complex systems, products and services, by multi-disciplinary teams embracing customers, partners and suppliers, working in concert to achieve a common aim to mutual advantage. Further, such developments are often multi-national and indeed frequently global in scope, and as such are ‘distributed’ in both graphical and temporal senses. In practice, such developments also involve many different design and analysis tools, both those supporting systems engineering directly, and those supporting the various specialist disciplines with which systems engineering interacts. Interoperability between tools within SE and with these disciplines is most desirable to increase quality, and reduce time and cost of the systems engineering process. The accepted way of achieving effective design data exchange across a wide variety of tools in an effective and economical way is neutral design data exchange. Such an approach, and its advantages and future potential have been explored in previous papers (Johnson et al. 2000, and references therein).

However, it is often difficult to ‘sell’ the case for tool neutral data exchange, or even simply the potential benefits of an underlying data model embracing the domain. At least one reason for this difficulty is the relative obscure relationship between process and data models. Process models are to a large degree intuitive and self-evident because they simply describe things that need to be done, and to some extent how these activities interrelate. The same cannot be said for
data models. Firstly, data models are much less familiar and understandable to the average practising systems engineer. Secondly, the relationship between data models and process models is neither straightforward nor particularly obvious. Thirdly, exploitation of a data model requires its realisation in one or more tools, and the relationship between systems engineering tools and the process is itself complex.

This paper attempts to shed light on the relationship of data and process models, and further to explain how the emerging Modular-AP233, when supported by systems engineering-relevant tools, will enable significant improvements in realising the systems engineering process. Following an explanation of the relationship between data and process models, the basic approach in this paper is to illustrate how the generic data model relates to one particular SE process view (ANSI/EIA-632). This is done both in a broad but shallow way, and also, by focussing in a specific area, in a deep, detailed way.

**Terminology.** Information models and data models are conceptually similar, in that they focus on concepts about which information systems (which can include both information technology and people) hold and manipulate data. One dominant way such models are constructed is around entities (real or virtual but tangible concepts), relationships between such entities, and attributes (details that distinguish one instance of a concept from another). For instance, a model of a personnel system would have concepts like *person* (perhaps with instances identified by *person1* and *person2*), relationships like ‘works for’, and attributes like *last-name*, *first-name* (for instance instance *person1* might have *last-name*=Smith, *first-name*=John). This is termed an entity-relationship-attribute model, or ERA model (Chen, 1976). For the purposes of this paper, information models are used to mean such models at a higher level abstraction, in other words recognisable to practitioners in that domain (in this case an information model of systems engineering would use terms recognisable to systems engineers). In contrast a data model is a model of these concepts appropriate to the level of implementing an information technology solution, and as such typically has many more entities and other details than would be recognisable to the practitioners in the domain of interest. The AP233 model is closer to a data model, due to its emphasis in being appropriate to support inter-tool communication. For the rest of this paper, the term data model will be used. However, the word information is used in various explanatory contexts, intended to reinforce the point that we are referring to data that carries meaning.

STEP (ISO 10303, ISO 1994; Kemmerer 1999) is the international STandard for Exchange of Product data, and is split up into a number of component parts, including a series of ‘Application Protocols’ or AP’s, each covering a specific industrial domain. AP233 is the Application Protocol specifically targeted at systems engineering. SEDRES and SEDRES-2 (Systems Engineering Data Representation and Exchange Standardisation) were two European research projects running 1996 through 2001 that both initiated the AP233 development, and have produced five versions of an earlier draft to AP233, termed SEDRES/AP233 in this paper (see www.sedres.com). The SEDRES/AP233 data model is a monolithic data model, of a style similar to the older parts of STEP, such as that covering 3D geometry, AP203. SEDRES/AP233 will be issued as an ISO Publicly Available Specification, PAS 20542. The newer STEP AP’s are organised following a ‘modular’ approach, hence the ongoing work to produce the modularised AP233, here termed Modular-AP233.

**Relationship between data and process models**

Much of the published literature, and indeed relevant standards, on systems engineering as a
discipline, focus on a process - or activity - view of systems engineering (IEEE, 1994; Stevens et al. 1998; ANSI/EIA, 1999; ISO, 2002). One notable exception is the publication (Oliver, 1997) which is relatively unusual in attempting to represent systems engineering from both process (activity) and information perspectives.

**Data models.** So what is a data model, and how does it relate to a process model? A data model is exactly that, a model of data. It captures concepts about which a system must be aware. Conventionally, concepts include terms such as entities (or classes), attributes and relationships. Entities correspond to the primary, or significant objects (real or conceptual) in a domain of interest. If the domain were a conventional book library, the concepts are likely to include: *book, customer, loan*. Attribute is one conventional term used to capture characteristics of the entities of interest. For the library example, *book* is likely to have attributes: *title, author, ISBN number, date of publication, publisher*. For *customer*, attributes would include: *first name, last name, address, phone number*. Finally, the data model captures relationships of importance. In this library example, relationships include: *book is-loaned-to customer* (equivalent to *customer has-loaned book*).

If one were developing a library system, it is likely that one would develop a functional view of the intended library, in addition to the data view. The functional model of this domain is a library process model. However, this is where the data view can directly reveal aspects that might otherwise remain hidden or more obscure in the functional view until we get later into the analysis process. Consider the following example. We know that a library needs to support an additional relationship that allows the customer to reserve a book, if the required book is not currently available. The additional important relationship is: *customer reserves book*. But of course while a customer may reserve book X, the customer actually removes from the library a particular (physical) instance of book X, a *book-copy*; there is a fundamental distinction between a *book* and a *book-copy*. The customer does not care which instance (or *book-copy*) he will take away, he simply wants one when one becomes available. So a data model often adds insight and clarity to a domain. Further the data model captures (in this example) not only the distinction between entities *book* and *book-copy* (with the respective attributes), but that these concepts are related in a particular way (*book has [many] book-copy*). In this way a data model adds rigour to the expression of concepts in a domain.

Finally a data model will capture more detailed information that characterises the 'business rules' that govern the domain. For instance, a customer would need to be known to the system (perhaps with all details), before being able to reserve or borrow a book. There may be a limit on the number of books that a person is allowed to borrow at a time. One would expect that a person's request to be removed from the system would not be allowed if it was recorded that at that time they still had books out on loan etc.

Such a conceptual data model is typically one of the earlier steps in developing information-intensive information technology systems. The conceptual schema are subsequently refined to data schema, which support the actual implementation, and are likely to be expressed in the language of the chosen implementation technology. Conceptual data models, as distinct to data schema, are essentially implementation independent, and are typically presented in a language that is not tied to any particularly vendor or data base management system. Examples of such languages would be the EXPRESS data modeling language used in STEP, or the class diagrams and supporting information of the Unified Modeling Language, UML (Booch et al, 1999).

**Process-data model relationship.** What is the simplest way to explain the interrelationship of data and process models? Simplistically, if we view the process view as a form of data flow
diagram, it is likely that many of the entities will map to the flows on the data flow diagrams. Typically many attributes will map to data terms embedded in flows or stores. The functionality represented by the functions on the flow diagrams supports both the creation, reading, update and deletion of the information represented by the flows (and in the data schema), and also supports the functionality implied by the business rules. Conventional data base systems have in-built functionality that may mean the functional support of business rules is a 'given', as a result of making use of that sort of technology. Figure 1 attempts to illustrate this interrelationship.

In summary:
- Data models offer insight
- Data models add rigour to the definition of concepts
- Data models underpin the capturing of business rules or invariants

Process and Data Models are complementary, representing different and ideally consistent views on the same domain. Representing a complex domain such as systems engineering from only a process perspective is like trying to communicate to a building contractor the design of a house by only providing plan and front elevation views, and no side view.

In subsequent sections we explain the anticipated scope of Modular-AP233, and summarise its current status, then focus on ANSI/EIA 632 as one representative expression of systems engineering, and the relationship of AP233 to this process standard.

### Modular-AP233: Status and Scope

The current planning view of Modular-AP233 is represented by the module sets indicated in Table 1. The anticipated Modular-AP233 embraces the content of SEDRES/AP233, and with additional areas. In the table *New* means that that module set is new to AP233, while *Extended* indicates that it is extended over what was in SEDRES/AP233. Figure 2 indicates a conceptual focus of AP233 against the highest level view of ANSI/EIA-632.

<table>
<thead>
<tr>
<th>Model area</th>
<th>Module Set</th>
<th>New to, or extended from SEDRES/AP233?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Definition</td>
<td>Text-based Requirements [TBR]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Property-based Requirements [PBR]</td>
<td>Extended</td>
</tr>
<tr>
<td></td>
<td>Structural Models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Behavioral Models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cost Models</td>
<td>New</td>
</tr>
<tr>
<td>Infrastructure &amp; System Definition</td>
<td>Validation and Verification Model</td>
<td>Extended</td>
</tr>
<tr>
<td></td>
<td>Allocation Model</td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Data Representation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Risk Analysis</td>
<td>New</td>
</tr>
<tr>
<td></td>
<td>Scheduling</td>
<td>Extended</td>
</tr>
</tbody>
</table>
Table 1: Planned Module Sets within Modular-AP233

At the time of writing, the TBR data model is at Beta release, PBR at Alpha release, while work has already started to scope and perform initial data modeling on other Module Sets, including Structural Models and Behavioral Models. The work on Behavioral Models is being done with close liaison to the OMG activity to develop support for SE in UML (see syseng.omg.org). Because UML 2.0 is itself under development, there is an opportunity to investigate (and to some extent influence) the support for behavioural modeling in UML. Prototypes of interfaces for the TBR module for a number of tools have already been produced, in one case by the respective tool vendor directly. Requirements, requirement traceability, and requirement allocation to a system hierarchy have been exchanged among these tools. Subject to successful bids for significant funding in both Europe and the US to enable the work to progress at a level similar to the SEDRES projects, it is anticipated that the full Modular-AP233 could reach maturity in 2004.

One SE Process View - ANSI/EIA-632 - and relationship to AP233

There have been a number of attempts to represent ‘the’ systems engineering process, each building on what has been done before, and in some sense tackling the domain from a different perspective and with different breadth and depth. Most notable published ‘standards’ documents, all from a process perspective, include: IEEE 1220, ANSI/EIA 632, ISO 15288. In this paper we concentrate on the relevance of AP233 to the support of one of these process representations, ANSI/EIA 632, because arguably it has the largest coverage of both breadth and depth of these standards, and because INCOSE contributed
significantly to its very development.

Figure 2 is the high level view of ANSI/EIA 632, which illustrates (informally) relationships between the processes of this Standard; we say 'informally’ as even this top-level view is not consistent with the next-level 'process' diagrams within the same standard. Corresponding to this high-level view of the SE processes, are a set of 33 'Requirements', which are relevant to a systems developer who (in the words of ANSI/EIA 632):

"should: (1) decide which of the processes in Figure 4a [Figure 2 in this paper] apply to their enterprise; (2) decide which requirements from this Standard apply for the processes selected; (3) establish appropriate policies and procedures that govern project implementation; (4) define appropriate tasks for each of the selected requirements; and (5) establish the methods and tools to support task implementation."

The expectation is that fulfilling each of these process Requirements by a developer applying ANSI/EIA 632 is mandatory, as illustrated by the use of a shall, for example:

**Requirement 16—System Technical Requirements**

The developer shall define a validated set of system technical requirements.

However, all SE Process requirements are similarly "high level" and are elaborated through a succession of should statements, for example the above Requirement is elaborated with these statements:

The developer should plan and do appropriate tasks to complete this requirement. Tasks to consider include the following:

a) Establish required transformation rules, priorities, inputs, outputs, states, modes, and configurations, as appropriate to each system product.

b) Define operational requirements to include operational profiles, and for each operational profile the utilization environment, events to which system end products must respond, frequency of use, physical and functional interfaces, and system functional requirements (what system end products must accomplish).

c) Define the performance requirement (how well each functional requirement must be accomplished), including identification of critical performance parameters

d) Analyze acquirer and other stakeholder requirements to define human factor effects and concerns, establish capacities and timing, define technology and product design constraints, define enabling product requirements, identify conflicts, and determine criteria for tradeoff analyses to resolve conflicts. (etc)

In the descriptions of activities within the text of ANSI/EIA-632 are explicit or implied references to the majority of the data concepts that are fundamentally important to systems engineering. Taking the fragment above, and driving out the referenced concepts, we see the content of Table 2, showing the implied information concepts and the (most closely related) AP233 entity. We can see a relatively close correspondence between the process / activity focussed representation of systems engineering in ANSI/EIA-632, its explicit (or implicit) concepts and the data modeling fragments within AP233. Note however the need for interpretation of the verbatim text in ANSI/EIA 632. For instance, what is 'stakeholder requirement'? A requirement about a stakeholder? A requirement expressed by a stakeholder? By
an individual stakeholder, or by an individual representing a group of stakeholders (maintainers, users, acquirers)?

The glossary of ANSI/EIA-632 provides further insight:

**stakeholder:** An enterprise, organization, or individual having an interest or a stake in the outcome of the engineering of a system.

**NOTES**

1. Examples of stakeholders are acquirer, user, customer, manufacturer, installer, tester, maintainer, executive manager, project manager, and all other personnel having a stake in the development or outcome of the engineering of a system. The enterprise as a corporation or agency and the general public are also stakeholders.

2. …

3. …

**stakeholder requirement:** A requirement that represents what stakeholders of a system need or expect of the system products.

<table>
<thead>
<tr>
<th>ANSI/EIA 632 text</th>
<th>Implied information</th>
<th>Associated Modular-AP233 Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Establish required transformation rules, …</td>
<td>transformation rules, priorities, inputs, outputs, states, modes, configurations, system product</td>
<td></td>
</tr>
<tr>
<td>b) Define operational requirements to include operational profiles, and for each operational profile the utilization environment, …</td>
<td>operational requirements, operational profiles, utilization environment, events, frequency of use, physical and functional interfaces, system functional requirements</td>
<td>Operational requirement Interface requirement Functional requirement</td>
</tr>
<tr>
<td>c) Define the performance requirement …</td>
<td>performance requirement, critical performance parameters</td>
<td>Effectiveness measure; functional requirement; temporal requirement</td>
</tr>
<tr>
<td>d) Analyze acquirer and other stakeholder requirements to define …</td>
<td>stakeholder requirements, [human factor effects and concerns, capacities and timing], technology and product design constraints, enabling product requirements, conflicts, criteria for tradeoff analyses</td>
<td>Requirement; Regularization function; Requirement_weighting_property</td>
</tr>
</tbody>
</table>

**Table 2: Analysis of ANSI/EIA-632 text for information concepts**

Most likely ANSI/EIA-632 *meant* a requirement expressed by a particular stakeholder group representative at a particular point in the development. And what about change? Of course requirements will be updated over time, to clarify and update the intentions and needs, so we will have requirement versions. There is clearly a complex set of interrelated concepts around the terms *stakeholders* and *requirements*. The combination of Requirements Module and Organizational Structure Module parts of the AP233 model would enable such interpretation to be made unambiguously: what is the requirement [text], who stated it, who do they represent, what is the version of the text, etc. This is an example of the way in which a data model such as AP233 can give rigour to the concepts often ambiguously stated in a natural language document.

It should be noted that although AP233 has been primarily driven by the need for support to the 'technical processes' in ANSI/EIA-632 (highlighted in Figure 2), it is also supportive of the non-technical processes Acquisition & Supply and Product Realisation. For instance, establishing increased formalism for expression of requirements can significantly ease the process of establishing an unambiguous agreement between Acquirer and Supplier, the
'agreement' being at the centre of the Acquisition and Supply processes.

**The role of AP233 in supporting the SE process**

Figure 3 below indicates conceptually a subset of the Modular-AP233 relevant to the area of focus above. The concept of **Requirement** and **Requirement_version** occur in a Requirements Module; these are built from the (existing) generic resources (including PDM Module) in STEP to support item identification, versioning, and relationship to Product. The **Requirement_view_definition** (derived from existing material) enables all requirements to be associated with a particular view on the product (a 'requirement' view).

Since text is only one of many forms of representation, the Text Representation Module relates to the **Representation** entity, again from STEP generic modules. **Text_representation_item** actually stores the text of a particular **Requirement_version** rather than of **Requirement**, as changes to the text description of a requirement are most commonly associated with the change to that version of the requirement, not to the requirement itself. Requirements may be classified via a **Requirement_classification** entity, to support various types, including functional, non-functional, interface etc.

As an example, a **Requirement** (of type Test Requirement) might be 'Thermal Balance Test'. The text of the most recent **Requirement_version** (actually held in an entity **Text_representation_item**) could be 'A Thermal Balance (T/B) test shall be conducted on each flight hardware system in the mechanical configuration(s) and electrical power mode(s) appropriate (operating or non-operating) for the mission thermal event being simulated....'.

**Practical exploitation.** Simply the existence of a data model to complement a natural language process standard does not itself provide anything other than insight and encourage rigour. The real exploitation occurs when the concepts are realised in information technology systems and in import / export interfaces of systems engineering-relevant tools that map the SE data to AP233 terms. A practical systems engineer is unlikely to need to know of the explicit terms within AP233; if a functional modelling tool uses the term **activity** for the AP233 concept of **function**, that mapping would be handled by the implemented import/export interfaces.

Given that we can then envisage a situation where many or all of the tools directly used by systems engineers have AP233 interfaces, we will then have relatively painless capabilities to move appropriate systems engineering data between the tools used by systems engineers in various business situations. Figure 4 illustrates several of these situations. On the left of the
figure identified as 'Through life-cycle phases' is the need to move design data through phases of the lifecycle, as designs mature and are subsequently maintained, or re-used. The context also embraces the implicit interaction between systems engineers and the 'specialist' engineering disciplines (electrical, electronic, safety, software, hydraulics, structures, support etc.). Of course, AP233 is not intended to support all requirements for product data exchange. In satisfying all needs for data exchange for a business requirement, many different parts of STEP (the different Application Protocols, AP203, AP212, etc) come into play.

One view of the interaction with different 'disciplines' can even be seen within one of the SE standards. Figure 5 is a reproduction of the earlier Figure 2 from ANSI/EIA 632, but highlighting the categories of tools that typically support the different processes. Clearly there are needs to exchange information with the tools that support these processes, although this requirement is not greatly elaborated in ANSI/EIA 632.

On the right of figure 4 is a different context for SE data exchange, identified as 'Between stakeholders'. Frequently, and not
unreasonably, different tools are used by acquirers of systems, to those used by the prime contract developers and to those used by the subcontractors and suppliers. It is likely that all three categories deal with requirements of some sort, however the needs of an acquirer are different to those of a developer, hence the focus on a different tool. It is even not uncommon for major partners on a development to (or at least want to) use different tools for, say system architecting, to reflect their own individual preferences or house standards.

Examples of AP233 supporting data exchange. Examples of the practical exchange of (subsets of) systems engineering data are available from the earlier prototyping of the SEDRES-2 project, presented at the Final Event in 2001 (see www.sedres.com). Examples available there focus on exchange of product requirements and functional / behavioural models, and illustrate the use of commercial tools such as Software-through-Pictures, Teamwork, Statemate, and MATRIXx. Although the model is now being re-architected and extended to become Modular-AP233, the practical examples and capabilities shown there will, at some point, also be possible with the final standard.

Included below are examples now available from Modular-AP233, for text-based requirements. Figure 6 illustrates example textual requirements data in the requirements management tool DOORS (Telelogic). Figure 7 illustrates the same data in the tool CORE (Vitech). (Vitech CORE should not be confused with the BAE SYSTEMS requirements /
A functional modelling tool also called CoRE, mentioned in SEDRES materials. Requirement identification codes and requirement text has been exported from DOORS to an AP233 structured file, and re-imported into CORE, preserving identification codes and requirement text. Less visible from the screen shots is the fact that requirement interrelationships are also preserved.

Relating this example back to the contexts illustrated in figure 4, it is often the case that an end-user acquirer, such as a defense agency would use a tool such as DOORS for their customer requirements specification process. In contrast, it is likely that a defense contractor would make use of a tool such as CORE for first-cut systems requirements elaboration and concept design analyses. Also (see figure 5), DOORS would be categorised by many to be a Requirements Management tool, while CORE would be identified as a Design Definition tool, so we can see that this example directly illustrates AP233-supported information exchange required by ANSI/EIA-632.

**Further possibilities for exploitation.** Clearly establishing a standards-based way of exchanging requirements between such tools, with the flexibility it gives to subsequent
in information reuse and also long term archiving, can lead to significant business benefits. However, one type of exploitation still remains to be fully investigated, and that is the use of AP233 for design consistency checking, making use of multiple exports and additional consistency checking utilities. It is easy to mistakenly think that AP233 is only about data exchange, that information matures on one tool and is then migrated to a subsequent tool. In reality of course, information matures across many tools (often reflecting the work of different teams) simultaneously, and a real challenge is maintaining all data consistent, at least at appropriate points in time or maturity gates. It is likely that the benefits of exploiting AP233 for design consistency checking and holistic design evolution far out-way its benefits for simplistic data exchange.

**SUMMARY**

This paper has attempted to explain the relationship between process and data models, and how data models complement process models, and can add rigour to the terminology and concepts of natural-language written process models. It has illustrated how specific concepts implied by a fragment of the ANSI/EIA-632 standard is reflected explicitly in a fragment of the Modular-AP233. It has briefly explained how the enabling of the various tools used by systems engineers with AP233 interfaces will enable the seamless, heterogeneous support for systems engineering processes, such as ANSI/EIA-632, to be realised. The result is better quality systems at reduced cost through rigorous and automatic exchange of information among tools.

**2006 UPDATE**

The basic thrust of this paper remains true; AP233 is seen as a significant enabler to supporting conceptual and practical systems engineering processes. Since the paper was originally written, the project has had difficulties in establishing significant funding in recent years but has nevertheless made further progress, particularly in reusing and building on the growing set of modules within the STEP Modules repository. The scope has remained essentially the same, albeit with some adjustments: recognition of a need to support business and functional rules, and support for Decision Making. There has also been some further tool interchange testing, particularly in the area of scheduling. State-based and function-based behaviour modules near completion, and the schedule anticipates release of a Draft 3 and further testing later in 2006. The slow but steady progress expects completion of Edition 1 to Committee Draft (CD), with subsequent Draft International Standard, and International Standard before 2008.

However, lack of large scale interest in AP233 remains surprisingly low. This low interest is in stark contrast to the high level of interest in the parallel development of the systems modelling language SysML (SysML, 2006), and interest apparent in both user and tool vendor communities. Therefore, two key challenges remain for the AP233 team:

- Convincing significant players of the unique potential of AP233, in concert with the rest of STEP, including PLCS (AP239), and in coexistence with SysML, to achieve full realization of SE aspirations such as:
  - Support for SE principles across tool boundaries
  - Working designs at multiple levels of abstraction
  - Raising visibility of complex SE projects, system designs, qualification etc.
  - Unifying technical development and engineering management (see Johnson, 2004)
  - Acceleration of the use of engineering metrics in the SE domain.
• Subsequently acquiring significant funding to complete the development, and get solid buy-in from both vendors and SE user organizations.

Activities are already underway to tackle the first major challenge by BAE SYSTEMS, first internally then externally, by tackling the underlying issue of why major SE organisations (or at least the fund-holders!) appear to have such a significantly low view of the value of product-wide systems engineering information, and the outcome is expected to be the subject of subsequent presentations and papers.

ACKNOWLEDGMENTS

The authors acknowledge the work of the SEDRES/SEDRES-2 project members, and AP233 team, in performing the work that has made the realisation of AP233 possible, and express thanks to Dr David Oliver for providing helpful comments on an earlier draft of this paper.

REFERENCES

ISO, ISO-15288, 2002
Oliver, Dr David, Engineering complex systems with models and objects, McGraw-Hill, 1997.
SysML, see http://syseng.omg.org/, 2006.

BIOGRAPHY

Dr Julian Johnson obtained a BSc in Physics and PhD in Physics (Space Science) at Kings and University College, London, respectively. He worked for over 8 years as a research scientist, at University of London, at NASA/MSFC, Alabama, and the University of Southampton, UK. He has experience in the management of software engineering projects and in the application of various systems analysis approaches. He has worked in BAE SYSTEMS since 1987, was the Technical Manager on SEDRES/SEDRES-2, is an executive scientist at the Systems Engineering Innovation
Centre (SEIC), and continues to be involved in AP233 development and other systems engineering standards. He is a member of the INCOSE UK chapter.

Dr Ian Bailey is a senior consultant with ModelFutures and has been working in the area of data integration for the last 10 years. Previous to this, Ian was a mechanical engineer in the automotive industry specialising in power-train design. He has a PhD in software integration for engineering data. Recently, Ian has been working with the UK MOD and the NASA Jet Propulsion Laboratory to develop AP233. Ian has been the technical lead for the development of AP233 interfaces to DOORS, Requisite Pro and Vitech Core. He has also advised on the integration of AP233 with the EDS Slate tool and 3SL's Cradle product. He is the designer of the high-level API for AP233, a tool which enables software developers to quickly work with AP233 files. He is a member of the INCOSE UK chapter.