Support for Object-Orientation in AP-233

Asmus Pandikow and Anders Törne
Real-Time Systems Laboratory
Department of Computer and Information Science
University of Linköping
581 83 Linköping, Sweden
E-Mail: {asmpa, andto}@ida.liu.se

Abstract. This paper is motivated by the need to bridge the gap between the engineering methods used in software engineering and those used in systems engineering. Different alternatives attacking the problem are presented and evaluated. An implementation of one of the alternatives is described, namely the inclusion of the most important object-oriented concepts from the OMG Unified Modeling Language (UML) into the working draft 5 of the ISO-10303-233 (AP-233) systems engineering standard proposal. These concepts have also been integrated into the AP-233 structures for version and configuration management and hence, allow now for the traceability of design elements between systems and software specifications. The extensions allow to include UML conformant tools in AP-233 controlled system designs and on the other hand, the use of AP-233 management capabilities for designing software.

INTRODUCTION

Object-oriented methods have become prevailing in software engineering. Despite the fact that their concepts might also be employed in other engineering disciplines, object-oriented methods have not yet been fully recognized outside the software world. The importance of software for contemporary systems is steadily growing, hardware and software are increasingly integrated, e.g. in embedded systems.

Currently there is a gap between systems and software engineering methods, as explained in (Cocks 1999). Developing such integrated systems, it is desirable to be able to reference both, hardware and software elements, in order to provide full traceability throughout the complete specification of a system. Such comprehensive specifications provide the means for system-wide management functionalities such as version and configuration management. A global specification also allows for system-wide traceability links between single specification elements, e.g. the allocation of requirements to systems as well as to single software constructs.

At present, such global capabilities supporting system-wide traceability, are being manifested in working drafts of a proposal for the international systems engineering standard AP-233[1]. This work is performed in the SEDRES-2 project, see (SEDRES-2 2000), and the AP-233 working group, see (ISO AP-233 Website 2001).

The need to integrate object-oriented concepts in the working draft in order to support modern software engineering methods with AP-233, origins from the SEDRES-2 project. The original intention, as stated in (Pandikow et al. 2000), was to integrate object-orientation and traditional structured methods in AP-233 at the level of single object-oriented respective traditional structured concepts in order to exchange specifications of both methods on design concept level. This turned out to be difficult, as for the greater part concepts of one method have no matching semantics in the other methods.

Nevertheless, integrating the concepts on a higher level, i.e. including concepts from both worlds in generic design management capabilities, would also allow to achieve the most important goals described in (Pandikow et al. 2000):

- Close integration of software engineering and systems engineering
- Change and configuration management support for the major UML concepts.
- Traceability of single elements between systems *and* software specifications
- Enable the use of object-oriented methods outside the software world
- Enable partly data exchanges between tools of different design methods

Object-orientation and Systems Engineering. As stated above, object-orientation has started to be recognized in the systems engineering community, as software becomes more and more important in the development of systems. Especially industries with software-intensive products, such as in telecommunication, employ object-oriented methods, but also other branches have started to use object-oriented methods for analysis and design.

Also, there are efforts towards creating object-oriented systems engineering methods, such as the

[1] AP-233: “Application Protocol 233”, part 233 of the ISO 10303 (STEP) standard

OOSEM, described in (Lykins et al. 2000). In (Axelsson 2001) the author extends the UML notation such that it can be employed for designing real-time computer systems. The author creates additional elements that allow for richer specifications of a physical architecture and for modeling continuous-time relationships.

Nevertheless, object-orientation is new, compared to traditional structured methods, which have been used and matured over decades. There is limited experience with systems that were in part or entirely developed with object-oriented methods. Also, the ways to integrate with the existing and legacy work in order not to lose past efforts is not yet solved. This paper starts to work on the harmonization of object-oriented and structured design methods in order to allow for an improved interdisciplinary communication among engineers.

Prerequisites. Reading this paper requires some knowledge about AP-233 and what it is aiming at, as well as some understanding of basic object-oriented concepts. However, appropriate links to secondary literature in the areas can be found in the text and references.

The examples in this paper are given in EXPRESS-G and UML notation. If needed, additional information about the EXPRESS language family can be obtained from (ISO 10303-11 1994). Also, a description of UML concepts can be obtained from (Muller 1997).

Paper Outline. After this introductory chapter, the UML is discussed with respect to the integration intentions. Then, AP-233 in its state before the integration is briefly discussed. A deeper understanding of AP-233 can be achieved by following the literature links in the chapter. After that, the approach taken to integrate the object-oriented concepts of the UML in AP-233 is described, followed by an example for the representation of an UML specification. Conclusions and a description of future work in this area end the paper.

THE UML

The UML, see (OMG UML v1.3 2000), has been selected by the authors for the integration efforts, because it provides a comparable view on software engineering as AP-233 on systems engineering. Other notations, such as Verilog or VHDL for hardware design, provide for the greater part only domain specific views and do not have the same potential for being employed in other engineering disciplines as the UML.

The UML has been adopted as standard by the OMG in 1997, see (OMG Website 2000), and contains all of the major object-oriented concepts, such as use cases, class diagrams, etc. It is currently the most favored and comprehensive notation for software engineering. The integration of the UML and AP-233 is difficult, as the UML is evolving in a different pace than AP-233. Currently, the UML 1.4 is being released, UML 2.0 is under development and supposed to be released in 2001, whereas AP-233 strives for a stable solution that is durable and unchanging for a longer period. Furthermore, the specification of the UML leaves the interpretation and representation of details of some of its elements to the software tools. Also, the semantics are not always defined with formal rigor and may result in ambiguous interpretations across different tools. This makes it difficult to integrate some UML concepts with existing counterparts of the AP-233. Hence, the proposed integration approach results in partly parallel constructs for structured analysis and object-oriented concepts, even when a closer integration may have been desirable.

The unambiguous definition of parts of the UML semantics is to be tackled by the OMG in future releases of the UML, see (OMG UML Roadmap), but will probably in details be left undefined in order not to constrain different interpretations of the UML. For software engineering this does not appear to hinder the dissemination of the UML, as it is already heavily employed. Nevertheless, fully defined semantics are vital for all engineering disciplines. Especially collaboratively developing heterogeneous systems requires unambiguous exchangeability of specifications. Nevertheless, the integration of the major object-oriented concepts found in the UML is still possible, as it is unlikely that the major concepts will change in their appearance and structure, but only be refined in their semantics. The concepts have proven their expressiveness and usefulness in the past years of object-oriented software development and can therefore be considered to be stable enough for the integration purposes.

The UML can be seen as collection of the major object-oriented notations. It has been developed on the basis of methods by (Booch 1994), (Rumbaugh et al. 1991) and (Jacobson et al. 1992) and is the result of merging the notations to a unified notation supporting the greater part of object-oriented development processes. Although the use of single concepts of the UML is not bound to a certain process, there is a common interrelation between the concepts that can be found in all processes using the UML:

- Use cases are the starting point
- Class diagrams are created from initially known architectures or may also evolve from refining use cases
- Behavior diagrams (collaborations, sequences, statecharts, and activities, see (Muller 1997) for more detailed descriptions) are refinements of single use cases (and may represent the behavior of class operations)
- Implementation diagrams are built on the basis of previously created specifications and existing physical architecture

The concepts can be viewed as major object-oriented concepts and have been chosen for being fully integrated in the AP-233 proposal.

STEP AND AP-233

The STEP[2] framework (ISO 10303) provides mature capabilities for traditional engineering disciplines such as mechanical and electrical engineering. The management capabilities of STEP standards, such as configuration management, are widely adopted in industry.

AP-233, as part of STEP, provides comprehensive support for the most important systems engineering concepts in the following areas:

- Requirements engineering
- Functional architecture
- Physical architecture
- Management information
- Inter-allocation of specification elements and traceability

As mentioned in the introduction, up to working draft 5 there was no support for object-oriented analysis and design concepts in AP-233. Hence, the support for modern software engineering was lacking and object-oriented software specifications could not explicitly be included in AP-233 compliant designs. More detailed information about the architecture of AP-233 can be obtained from (Herzog and Törne 2000).

ANALYSIS

The gap between the different engineering methods for software and systems as described by (Cocks 1999) has developed from employing diverging design paradigms in the respective engineering disciplines. While structured methods still prevail in systems engineering, has software engineering developed and employed object-oriented methods. The integration of both paradigms is not straightforward, as for many object-oriented design concepts no semantically matching structured concepts exist, and vice-versa. Nevertheless, the integration of both methods could be achieved in several ways:

1. Creation of a new unified software and systems engineering standard
2. Inclusion of systems engineering concepts in a software engineering standard
3. Inclusion of software engineering concepts in a systems engineering standard

[2] STEP: "Standard for the Exchange of Product Model Data", also "ISO-10303", a standardization framework by the International Organization for Standardization, ISO (see www.iso.ch)

The first alternative can currently be viewed as theoretical and impracticable, as both communities (software and systems engineering) each already undertake great efforts towards creating respective domain-specific standards, see (OMG UML v1.3 1999) and (ISO AP-233 Website 2001). Another completely new approach with both parties as stakeholders would probably not be acceptable.

The second alternative would mean major changes to existing software engineering standards, e.g. the UML, and is also not acceptable as the UML intentionally only represents the object-oriented approach. A viable solution in line with this alternative would be to create a UML systems engineering profile that tailors the UML to the needs of systems engineering, with the disadvantage of a distributed and unsynchronized standard (AP-233 together with UML profile).

Including software engineering concepts in the ongoing efforts towards creating the AP-233 systems engineering data exchange standard is currently the most promising alternative. AP-233 represents commonly agreed concepts and provides the right opportunity to harmonize with object-oriented concepts. Besides representing the most important concepts of the current systems engineering practice, working draft 5 of AP-233 also provides mature capabilities for managing models, versions and configurations. These are capabilities that UML specifications will gain from, as they are not natively supported by the UML.

The chosen alternative may not be approved by all members of the systems engineering community to be the best possible solution, as there may be different opinions about which alternative to prefer and at which level the integration shall take place. Nevertheless, the chosen approach can be seen as one of several implementations of the alternatives and can be taken to be compared to other future solutions.

OBJECT-ORIENTATION IN AP-233

When discussing AP-233 and its extensions in order to also capture object-oriented concepts, this paper actually refers to the information model that has been created by the SEDRES-2 project. The SEDRES-2 information model has been forwarded to the AP-233 working group and is updated since then by proposals from both communities. For simplicity reasons, this paper does not distinguish SEDRES-2 and AP-233 work, but the reader has to keep in mind, that the request for the inclusion of object-oriented concepts origins from the SEDRES-2 participants.

As described above, the UML has been selected as basis for the efforts of identifying the major object-oriented concepts. As a result of the different semantics of UML respective AP-233 concepts, the integration has been performed by including object-oriented concepts in the management capabilities of AP-233 rather than on concept level.

The overall goal was to create support for all UML concepts, preserving their characteristics and include them in the existing generic management capabilities of AP-233. For the greater part, the concepts have been introduced as new AP-233 concepts, except for the cases where AP-233 already offers semantically adequate alternatives.

The intentions in (Pandikow et al. 2000) to obtain a close mapping between object-oriented and AP-233 concepts have not been fully realized. Instead, the focus was put more on the integration of object-oriented concepts in the AP-233 version and configuration management capabilities rather than performing the integration on a lower level. Thus, the lower level integration, i.e. the mapping between single object-oriented concepts and their structured analysis counterparts, has only been performed in cases where the concepts are congruent or where AP-233 concepts had only slightly to be changed.

The greater part of the UML meta-model has been implemented as specified in the UML. The following paragraphs deal each with a certain area of the UML, describing its respective integration and possible peculiarities of the implementation.

For more detailed descriptions of UML concepts, see (OMG UML v1.3 1999) and (Muller 1997). Descriptions of basic object-oriented concepts such as classes, associations, etc., can be obtained from (Balzert 1999), (Muller 1997) or equivalent introductory literature.

Modeling Style. AP-233 is created within the ISO-10303 (STEP) framework and is modeled with the object-flavored EXPRESS language family. The STEP modeling style, as described in (Herzog and Törne 2000), based on principles from (Gregory and Reingruber 1994), strives for a flat super- / subclass hierarchy in order to serve the standardization process. Complex inheritance hierarchies are avoided. However, the UML meta-model heavily employs inheritance (which is not unusual for object-oriented designs) in order to reduce specification redundancy and get leaner modeling elements that are not overloaded with high-level features.

The approach taken to combine the conflicting objectives, is to decide for each modeling element, whether the considered feature is necessary for the model element or not. This results in fewer and less complex entities and relationships compared to the approach of attaching explicitly all inherited features directly to the model elements. Furthermore, the inheritance relationships have been implemented as flat EXPRESS sets (select statements) of modeling elements, where a set can contain another set, speaking object-oriented: the superclass is a set of subclasses.

Model Management. Managing versions and configurations in software engineering is usually performed as an external task external of the development process. Likewise, also the current version 1.3 of the UML provides a snapshot view on a design, i.e. it does not explicitly support different versions and configurations of a design. To achieve this, the existing model management capabilities of AP-233 have been adopted and the respective sets of version and configuration managed concepts have been extended by object-oriented concepts of interest. Furthermore, the capabilities of managing systems and subsystems have been extended and an additional mechanism for coupling UML diagrams to the AP-233 notion of systems has been created. Also, the interrelations between diagrams (e.g. one diagram is a specialization of another) has been made explicit. Figure 1 gives an overview on the EXPRESS implementation for this.

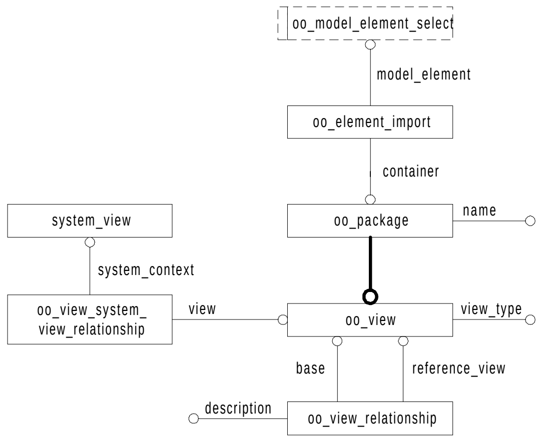

Figure 1: Managing Object-Oriented Models in AP-233

The *oo_view* entity represents object-oriented diagrams. The type of diagram (class diagram, use case diagram, etc.) is stored in the *view_type* attribute. Interrelations between diagrams are represented by the *oo_view_relationship* entity and are textually described in the *description* attribute. The *oo_view* is a kind of *oo_package*, the thick line in figure 1 between *oo_view* and *oo_package* depicts an inheritance relationship. Finally, the connection between elements on a diagram and the diagram itself is established through the *oo_element_import* entity, *oo_model_element_select* represents the set of possible diagram elements.

Global Concepts. The stereotype and constraint concepts have been included as specified in (OMG UML v1.3 1999). The UML properties are similar to the UML tagged values and are currently represented through the AP-233 entities for representing assessments of system characteristics. In future versions, AP-233 properties might be employed for this purpose. The UML notes are semantically equivalent to AP-233 justifications, hence a new concept has not been introduced for those.

Static Structure. The complete set of UML elements describing a system’s static structure, e.g. classes (with attributes and operations), objects and associations among them, have been added. For the greater part of static structure concepts, new entities had to be created, because there were no semantically equivalent concepts available in AP-233.

Figure 2 shows a simplified model of how the relationship between a class and its constituents (attributes and operations) is realized (note that most of the attributes have been omitted in this example).

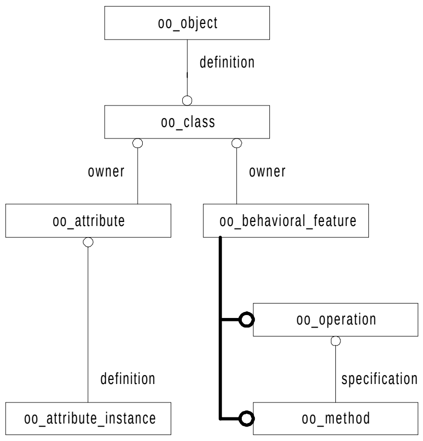

Figure 2: EXPRESS Example for Static Structure Elements

The relationship is implemented as in the UML meta-model, i.e. the owner attributes of *oo_attribute* and *oo_behavioral_feature* are connected to the set of UML classifiers (and attributes and operations are classifiers).

The example shows that a class may have attributes and operations assigned. Entities that represent instances of object-oriented concepts (e.g. objects are instances of a class) have been modeled explicitely and are directly related with the respective definition. This deviation from the UML meta-model, where all instances inherit from a common instance, has been made in order to flatten the inheritance hierarchy but is otherwise equivalent to the UML meta-model.

Use Cases. Use cases have no direct semantic equivalent in AP-233. The AP-233 concept of "partial system views" is partly equivalent, but as use cases are a prominent concept of their own and tightly coupled to other object-oriented elements, this would impose major changes in the system view concepts, which in turn would complicate the non-object-oriented use of the respective entities. Therefore, new entities for capturing use cases have been added.

Behavior. New entities representing the UML concepts for collaborations, interactions and message sequences have been added to AP-233. Note that especially sequence diagrams contain a number of elements that have no direct representations in the UML meta-model, because their existence is implicit, due to the existence of other elements.

As an example, figure 3 shows an UML sequence diagram that describes a simple communication between a sender and a receiver. The concepts *lifeline* and *activation* are vital elements of a sequence diagram, but are not explicitly represented in the UML meta-model. Hence, they are not in represented in AP-233 as well. Their existence can be derived from the existence of other elements: in a sequence diagram a *role* of a *classifier* has always a *lifeline* and, a *classifier* can only accept *messages* when it is "activated", i.e. it is has an *activation* segment. Such implicit elements are supposed to be created by software tools that implement the UML.

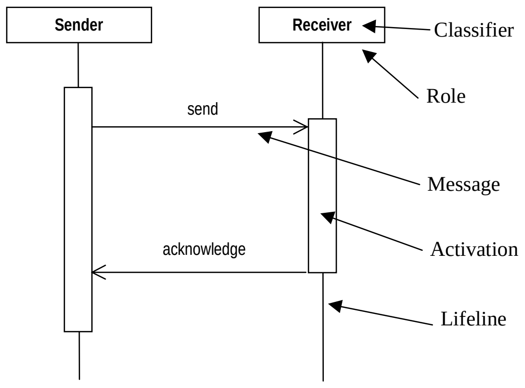

Figure 3: Sequence Diagram Example

The UML statecharts are represented by the AP-233 finite state machines. The major exception is the notion of "synchronized states" in the UML, which allows to synchronize concurrent states, but this can be substituted by modeling the respective statechart differently, i.e. creating states of explicit synchronization in the finite state machine.

The semantics of the UML activity diagrams can be represented by AP-233 causal chains. Hence, there have been only a few extensions made to AP-233 to support activities.

Implementation. The UML concepts for the representation of physical architecture is tailored to the needs of software engineering, i.e. is limited to computing devices. The AP-233 capabilities for the representation of physical architecture are far richer. Hence, AP-233 has in this respect only been extended in order to capture the UML notion of components and their relationships. Physical computing devices (the UML notion of "nodes") are represented by existing elements of the AP-233 physical architecture.

EXAMPLE

The example in figure 4 shows how an UML specification would be represented with the new object-oriented capabilities of AP-233 that have been described in this paper.

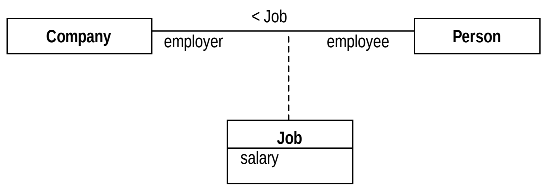

Figure 4: Class Diagram Example

The example contains a simple class diagram that describes the relationship between a company and its employees. There are two classes, *Company* and *Person*, and an association among them, the association *Job*. Additionally, the association class *Job* describes the association in more detail, i.e. whenever the association *Job* is instantiated, also an instance of the association class *Job* is instantiated.

The example would be represented by AP-233 as shown in figure 5 (note that entities with a black triangle in the bottom right corner depict an instance of an entity, italics are denoting attribute values).

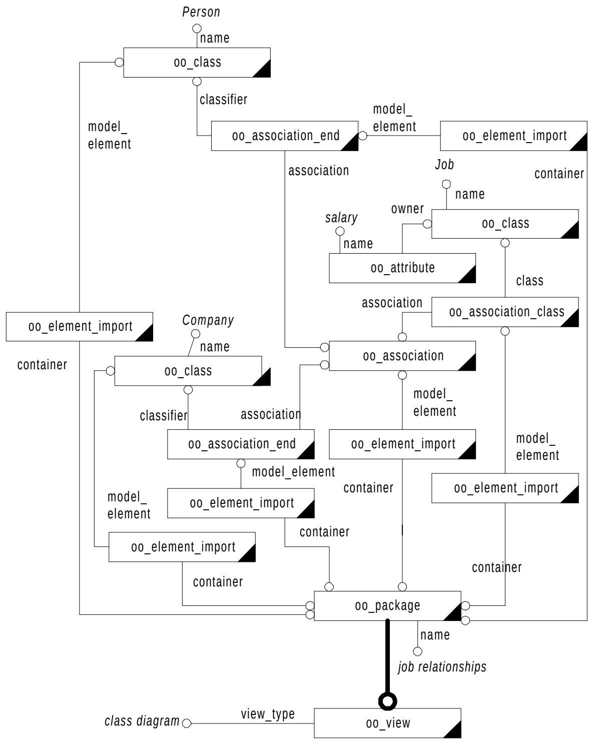

Figure 5: Example AP-233 Instance Representation

The class diagram itself is represented by an *oo_view* object and its superclass *oo_package*. The example diagram is entitled "job relationships", see *oo_package*. The classes *Company* and *Person* and are represented by respective *oo_class* objects, the association *Job* is represented by an *oo_association* object. The end points of the association (*employer* and *employee*) are represented by *oo_association_end* objects. The association class *Job* is represented by an *oo_association_class* object, connected to the *oo_class Job* and the respective *oo_association Job*. The attribute *salary* is an *oo_attribute*, attached to the *oo_class Job* through its attribute *owner*.

CONCLUSIONS AND FUTURE WORK

Possible approaches to bridge the gap between object-oriented software engineering methods and structured systems engineering methods have been identified and evaluated in this paper. The implementation of one of the approaches, namely to include software engineering concepts in the ISO-10303-233 standard proposal in order to support object-oriented concepts, has been presented.

Future reviews of this proposal will show, whether the chosen approach is appropriate and to which detail object-orientation is to be represented in AP-233. However, the integration of object-oriented software engineering concepts in AP-233 allows for the inclusion of modern software engineering concepts in the generic management capabilities of AP-233. Hence, UML compliant software specifications may now become part of global AP-233 controlled system specifications. Also, the other way round, AP-233 capabilities may be employed for software specifications.

The current efforts in the systems engineering community on evaluating the UML and formulating usability requirements for future UML versions in cooperation with the OMG indicates that there is great interest in the area.

The principal author intends to continue working on further implementations of the mentioned alternatives in order to enable the use of the UML also in other engineering disciplines than in software engineering. This will probably include contributing to a UML profile tailoring the UML to the needs of systems engineering. The results of this work may improve interdisciplinary communication among engineers, just what this paper is aiming at.

ACKNOWLEDGEMENTS

The authors would like to thank the participants of the SEDRES-2 project for their hard work. Especially, the valuable contributions from Erik Herzog, see (Herzog and Törne 2000), are gratefully acknowledged. Also, the financial support from the European Commission for the SEDRES-2 project is gratefully acknowledged.

REFERENCES

Axelsson, J.: "Unified Modeling of Real-Time Control Systems and their Physical Environments Using UML" in the proceedings of the "Eighth IEEE International Conference and Workshop on the Engineering of Computer Based Systems", IEEE Computer Society, 2000

Balzert, H. "Lehrbuch der Objektmodellierung", Spektrum Verlag, 1999.

Booch G.: "Object-Oriented Analysis and Design with Applications", The Benjamin / Cummings Publishing Company, 1994.

Cocks, D.: "The Suitability of Using Objects for Modeling at the Systems Level" in the "Proceedings of the Ninth Annual International Symposium of the International Council on Systems Engineering", pages 1047-1054, INCOSE, 1999.

Gregory, W. and Reingruber, M.: "The Data Modelling Handbook: A Best Practice Approach to Building Quality Data Models", Wiley and Sons, 1994

Herzog, E. and Törne, A.: "AP-233 Architecture" in the "Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering", INCOSE, 2000.

ISO 10303-11: "Industrial automation systems and integration - product data representation and exchange - part 11: Description methods: The express language reference manual", Technical Report ISO 10303-11:1994(E), ISO, Geneva, 1994.

ISO AP-233 Website: Internet homepage of the AP-233 Systems Engineering Working Group at http://www.sedres.com/ap233/sedres_iso_home.html, hosted by the SEDRES project, 2001

Jacobson I., Christerson M., Jonsson P., Övergaard G.: "Object-Oriented Software Engineering - A Use Case Driven Approach", Addison-Wesley, 1992.

Lykins H., Friedenthal S, Meilich A.: "Adapting UML for an Object Oriented Systems Engineering Method (OOSEM)" in the "Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering", INCOSE, 2000.

Muller, P.-A.: "Instant UML", Wrox Press Ltd., 1997.

OMG UML Roadmap: "UML 2.0 Roadmap Recommendations", OMG FTP Server at http://cgi.omg.org/cgi-bin/doc?ad/2000-06-01, OMG, 2000.

OMG UML v1.3: "OMG Unified Modeling Language Specification, version 1.3, first edition", OMG FTP Server at ftp://ftp.omg.org/pub/docs/formal/00-03-01.pdf, OMG, 1999.

OMG Website: Internet homepage of the Object Management Group at http://www.omg.org, OMG, 2000.

Pandikow A., Herzog E. and Törne A.: "Integrating Systems and Software Engineering Concepts in AP-233", in the Proceedings of the 10th Annual International Symposium of the international Council on Systems Engineering, pages 831 - 837, INCOSE, 2000.

Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.: "Object-Oriented Modeling and Design", Prentice Hall, 1991.

SEDRES-2: Project homepage of the "Systems Engineering Design Representation Exchange Standard 2" project IST 11953 at http://www.sedres.com, 2000.

SEDRES-2 AP-233 proposal: "AP-233 Working Draft 5", SEDRES-2 Website at www.sedres.com/documents/sedres_all_documents.html, SEDRES-2 project, 2001