Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1997 Proceedings

Americas Conference on Information Systems (AMCIS)

8-15-1997

Introducing Object Technology through the Use of the OPEN Methodology

B. Henderson-Sellers
Swinburne University of Technology

H. Younessi
Swinburne University of Technology

Introducing Object Technology through the Use of the OPEN Methodology

B. Henderson-Sellers and H. Younessi
Swinburne University of Technology

1. Introducing Object Technology

The key concepts of object technology include abstraction, modularization, information hiding and polymorphism. The key areas of object-oriented (OO) software development include architecture, reusability, iterative development. The key benefits are higher quality software (particularly when using a responsibility-driven approach incorporating the contracting metaphor), faster time to market, a much closer relationship with the customer/end user.

Object technology obviously focusses on "object" as the key concept. Although the word "object" covers a multitude of sins, at this stage we do not want to enter into discussion regarding the correct use of the words object, type, class, role and instance. In OPEN (see Section 2), they are collectively referred to as "CIRT" - which stands for class, instance, role or type. Generally, we can just use the word "object" generically.

Whilst there are a number of good introductory texts on object technology many of them have a technical and/or an industry focus. Few deal with issues pertinent to information systems - some examples here are Taylor (1990) and Henderson-Sellers (1997). None successfully deal with the undergraduate information systems student on their first encounter with object-oriented information systems.

Object technology has a number of faces: from technical to information systems management; from software engineering process to analysis and design model building. A coherent strategy either for adoption in industry or for incorporation into new subject curricula in teaching establishments is to use a single, fully comprehensive object-oriented methodology, such as OPEN (see below). The model building tasks and techniques form the basis of a course in OOAD/model building; the project management framework can be used to study organizational and management issues and the underpinning process can be a main thrust in a software engineering curriculum. Specialist topics such as metamodeling or distributed objects are also encompassed within the OPEN approach and can be taught separately for more specialized audiences.

In the following, we describe the OPEN methodological framework (Section 2), followed by a brief discussion of some of the advantages to business and educational programmes (Section 3).

2. Introducing OPEN

OPEN (standing for Object-oriented Process, Environment and Notation) is essentially a framework for third generation object-oriented analysis, developed by the (currently) 31 international members of the OPEN Consortium. As well as assisting in the building of object models of the problem and the solution, OPEN also supports business process modeling, offers guidelines on business issues, migration strategies and supports links to human relations issues. A prime concern of OPEN is software quality, the use of metrics and the business and educational environments.

In addition to synthesizing the best ideas and practices from MOSES (Henderson-Sellers and Edwards, 1994), SOMA (Graham, 1995) and Firesmith (1993), OPEN also utilizes concepts from BON, Mainstream Objects, Martin/Odell, OBA, RDD, ROOM, Syntropy, UML and others.

OPEN extends the notion of a methodology by including a process model (Henderson-Sellers et al., 1997) together with guidelines for constructing versions of this model tailored to the needs of individual organizations and problem types. The process model is an object model and, as such, is adaptable. One sample realization of this object model is the contract-driven life-cycle (Figure 1) suitable for rapid application development within an MIS context, but other instantiations are not precluded.

The key elements of OPEN are

- lifecycle process or (meta)model
- techniques for building objects models, for project management, for quality assurance etc.
- representation (including visualization techniques)

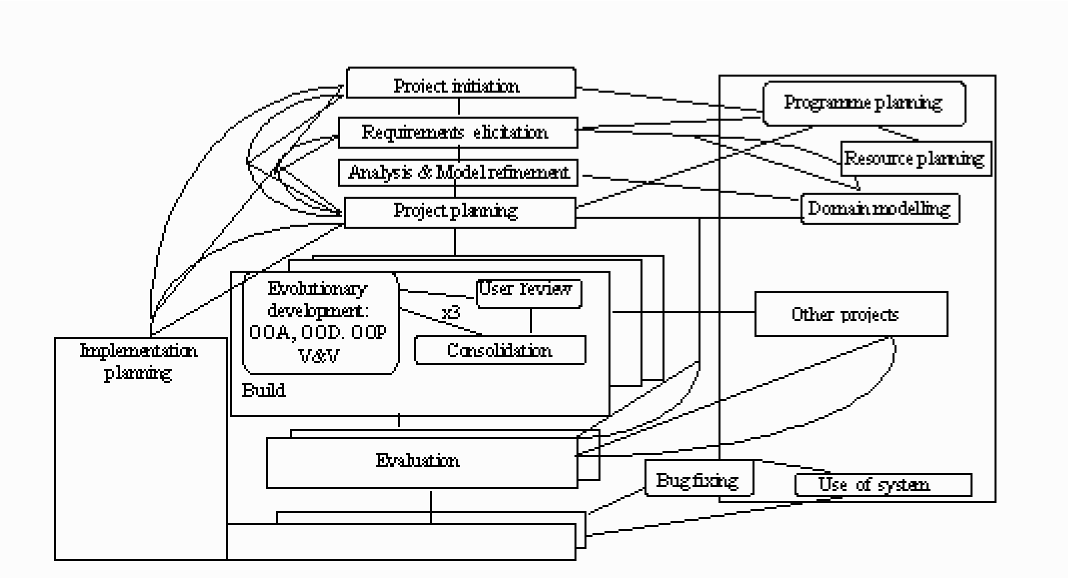
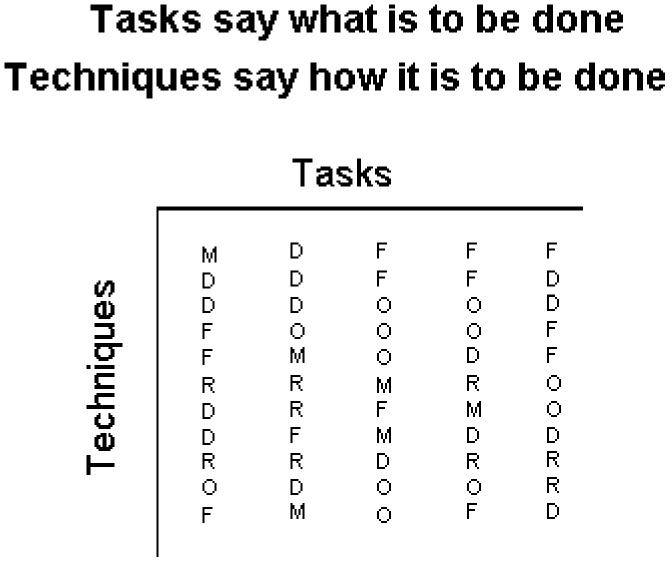


Figure 1 Contract-driven lifecycle process model (modified from Henderson-Sellers, Graham et al., 1996)


Tasks say what is to be done
Techniques say how it is to be done
Tasks
Techniques

M	D	F	F	F
D	D	F	F	D
D	D	O	O	D
F	O	O	O	F
F	M	O	D	F
R	R	M	R	O
D	R	F	M	O
D	F	M	D	D
R	R	D	R	R
O	D	O	O	R
F	M	O	F	D

Figure 2 The core of OPEN is a two-dimensional relationship betweenTasks and Techniques. For each task, a number of techniques may be useful. For each combination of Task and Technique, an assessment can be made of the likelihood of the occurrence of that combination. Some combinations can be identified as

mandatory (M), others as recommended (R), some as being optional (O), some as unlikely/discouraged (D) and other combinations that are strictly verbatim (F = forbidden).

The Activity objects of the lifecycle of Figure 1 consist of Tasks, activated by messages. An Activity is complete when its postcondition is satisfied and the developer can move to any new Activity by proving satisfaction of the precondition of the new Activity object.

Tasks can be considered the smallest unit of work within OPEN. Tasks are carried out by agents (people) using techniques. The appropriate Technique is selected by way of a two-dimensional matrix (Figure 2) which links the Task (which provides the statement of goals i.e. the `what') to the Techniques (which provide the way the goal can be achieved i.e. the `how'). Techniques range across project management, inspections and so on through to detailed theories and practices for requirements engineering and system modeling (Graham et al., 1997).

Finally, OPEN embodies a set of (object-oriented) principles which should be adhered to. These include:

o object modelling as a very general technique for knowledge representation

o encapsulation

o polymorphism together with o clear, jargon-free and well-founded definitions of all terms

o extensive use of abstraction techniques, a foundation for semantically cohesive and encapsulated "objects"

OPEN supports both the UML notation (Rational, 1997) and the OML notation (COMN) (Firesmith et al., 1997) and anticipates endorsing the OMG metamodel when determined (probably September 1997).

For more information on OPEN, the OPEN home page is located at URL:
http://www.csse.swin.edu.au/cotar/OPEN/OPEN.html

with mirrors at

USA: http://www.markv.com/OPEN

UK: http://www.scism.sbu.ac.uk/cios/mirrors/OPEN

Israel: http://oop.cs.technion.ac.il/OPEN

3. Business and Educational Advantages

One of the greatest technical advantage of an OO lifecycle process is that of (almost) seamlessness. "Objects" are used as the basic concept and unit of consideration during requirements engineering, systems analysis and design and in the OO programming language and future maintenance. "Objects" are the common language of users as well as technical developers.

One of the great educational advantages is also the seamless nature of object technology since it minimizes the number of "paradigms" or models that the students have to study and inter-relate. In addition, since object modeling is in some ways a superset of data modeling, learning to work with objects encompasses the older approaches with ER modeling and use cases and task scripts subsume the older DFDs. The focus on reuse means that libraries of objects and clusters of objects are available so that the work needed by the student to build a working system is considerably less; thus achieving a feeling of accomplishment earlier.

It is clear that OT in both business and education, particularly in the area of information systems, is growing rapidly. It is important that a coherent strategy for moving to this new and exciting paradigm is adopted. One such approach, focusing on a single unifying methodological framework, has been exemplified here.

References

Firesmith, D.G., 1993, Object-Oriented Requirements Analysis and Logical Design: A Software Engineering Approach, J. Wiley and Sons, New York, 575pp

Firesmith, D., Henderson-Sellers, B. and Graham, I., 1997, OPEN Modeling Language (OML) Reference Manual, SIGS Books, NY, 271pp

Graham, I.M., 1995, Migrating to Object Technology, Addison-Wesley, Wokingham, UK, 552pp

Graham, I.M., Henderson-Sellers, B. and other members of the OPEN Consortium, 1997, *OPEN's Toolbox of Techniques*, Addison-Wesley (in preparation)

Henderson-Sellers, B., 1997, *A BOOK of Object-Oriented Knowledge. An introduction to object-oriented software engineering* (2nd edition), Prentice Hall, New Jersey, 253pp

Henderson-Sellers, B. and Edwards, J.M., 1994, *BOOKTWO of Object-Oriented Knowledge: The Working Object,* Prentice Hall, Sydney, 616pp

Henderson-Sellers, B. and Graham, I.M. with additional input from C. Atkinson, J. Bezivin, L.L. Constantine, R. Due, R. Duke, D. Firesmith, G. Low, J. McKim, D. Mehandjiska-Stavrova, B. Meyer, J.J. Odell, M. Page-Jones, T. Reenskaug, B. Selic, A.J.H. Simons, P. Swatman and R. Winder, 1996, *OPEN: toward method convergence?* IEEE Computer, 29(4), 86-89

Henderson-Sellers, B., Graham, I. and Younessi, H., 1997, *Life Cycle Patterns*, Addison-Wesley, UK (in press)

Rational, 1997, UML Semantics, version 1.0, 13 January 1997 (unpubl), available from http://www.rational.com

Taylor, D., 1990, Object-Oriented Technology*: A Manager's Guide,* Servio Corporation, Alameda, CA, USA, 146pp