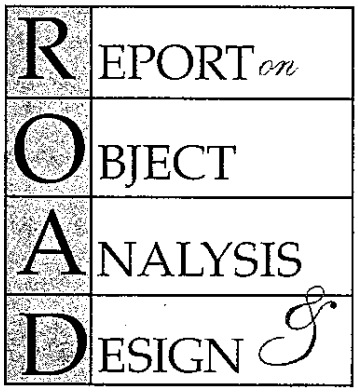
R|O|A|D

B. Henderson-Sellers

Ian M. Graham

Donald G. Firesmith

Methods unification: The OPEN methodology



It is clear that there are currently too many OO development methods and many (but not all!) of the differences between them are more cosmetic than substantive. Although competition may be fun, it soon becomes clear that, to those of us who really believe that object technology is "a good thing," the more important goal should be to create an environment in which the majority of software developers can move safely from their traditional development environments into the new age of object technology.

Consequently, most of the world's leading methodologists have been actively pursuing a goal of method unification. This led, in part, to the research work of the COMMA project on which I have been reporting here in the ROAD section of JOOP during the past year or so. The aim of COMMA, in a nutshell, has been to

- produce a metamodel of the leading OO methodologies
- extract from these metamodels ideas from which to construct a "core metamodel"—this was discussed by Henderson-Sellers and Firesmith[1]
- deliver this core to the OMG via the OADTF's RFP. This has been accomplished in terms of its major contribution to the metamodel submitted by the OPEN Object Alliance via Platinum in January 1997[2]
- permit this core model to be common to all methodologies while permitting extensions beyond the core so that each methodology can be directed at its own market

Although the last point is still to be fully realized, there are two unification projects that have influenced and been influenced by all this effort on metamodeling. In the first, Grady Booch and Jim Rumbaugh collaborated to create the Unified Method, Version 0.8, which later became the Unified Modeling Language (UML) Version 0.9 (and onwards). UML also includes contributions from Ivar Jacobson. UML consists of a notation and an underlying metamodel.

The second unification effort started with the announcement of a merger between Ian Graham's SOMA and Brian Henderson-Sellers' MOSES. Then the Firesmith method was merged in. OPEN, the method this merger creates, thus replaces three other methodologies at a stroke. In addition to these three, there are 28 other OO proponents in the OPEN Consortium dedicated to producing the OPEN methodology.

OPEN consists of a full life cycle process-centered methodology with wide-ranging emphases on reuse, quality, organizational issues including people, project management, and so on. It too has a metamodel (extended COMMA)[3] and a notation known as COMN (Common Object Modeling Notation) (see Fig. 1), which are collectively known as OML or the OPEN Modeling Language.[4] (COMN will be the topic of a future ROAD column.)

THE HEART OF OPEN OPEN is a full life cycle, process-focused methodology that supports all the elements of a methodology that one would expect:[5]

- a collection of rules and guidelines

B. Henderson-Sellers is professor of computer science in object technology at Swinburne University of Technology, Melbourne, Australia. He may be contacted at brian@csse.swin.edu.au. Ian Graham is Chairman of Graham Associates. He may be reached at 101710.3061@compuserve.com. Don Firesmith is a Senior Member of Knowledge Systems Corporation. He can be reached at dfiresmith@kscary.com.

- a full description of all deliverables
- a set of techniques and tools
- a set of appropriate metrics, standards, and test strategies
- a description of the underlying models for product and life cycle, i.e., process
- identification of organizational roles, e.g., business analyst, programmer

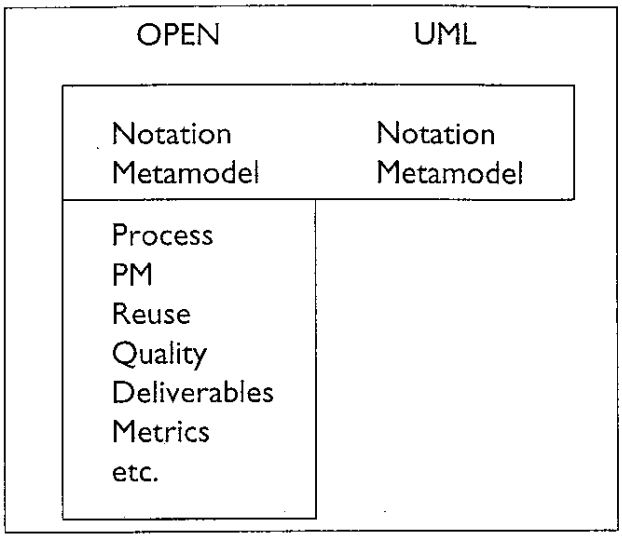


Figure 1. A method contains many elements. Only two of these are found in UML: notation and metamodel.

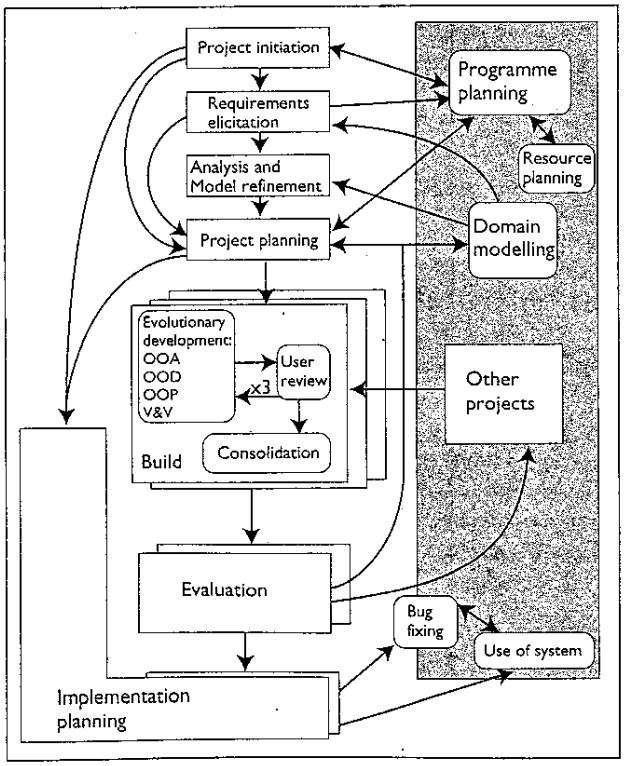


Figure 2. Contract-driven process life cycle model (adapted from Hendersen-Sellers *et al.*[6]).

- guidelines for project management and quality assurance
- advice on library management and reuse

The architecture of OPEN, at the metalevel, is of a set of life cycle Activities represented as objects (see Fig. 2). These objects have contracts with each other so that the flow of control can pass between these objects in any order so long as the contracts are met. This gives the necessary flexibility and tailorability to the overall architecture. Each Activity has a number of associated Tasks that describe what is to be done. These Tasks are the services/responsibilities/operations of the Activity objects. How the goals specified in these Tasks are achieved is described by a set of Techniques.

The key is how to choose the appropriate Techniques to fulfill the Tasks. This is accomplished by the use of a probabilistic matrix (see Fig. 3) that gives common, likely, rare (etc.) couplings between them. In fact, we specify the links with a probability selected from one of five levels: M (mandatory), R (recommended), O (optional), D (discouraged), and F (forbidden). The values in this matrix can either be determined as "industry averages" or they can be calculated at lower levels such as industry sectors (e.g., banking) or for your own particular organization or department. Preferably, once determined, the values should be retained from project to project, assuming the project characteristics are not vastly different.

OPEN ACTIVITIES Graham[8,9] discusses an embryonic, fully "object-oriented" life cycle that is shown in its most up-to-date form in Figure 2. Each Activity is shown as a box, either with rounded corners (an unbounded Activity) or with rectangular corners (Activity tightly bound in time). Because we are modeling these activities as objects, we can associate contracts with each Activity object (hence the life cycle name "contract-driven").

These are expressed primarily by pre- and post-conditions and by invariants; in other words, constraints that have to be met before an Activity can be commenced and final conditions that have to be met (and signed off) before another Activity can be initiated (triggered). Activities include well-specified testing activities as an integral part of the exit conditions and should deliver test results against both task scripts and a technical test plan of the normal kind.

It is important to stress that the progression order is neither prescriptive nor deterministic. Rather, the contract-driven life cycle permits:

1. ordering in an iterative, incremental and parallel fashion and
2. the tailoring by a given organization to its own internal standards and culture.

We indicate in Figure 2 some of the likely routes between Activity objects. But remember, the main governing constraint on the routes are whether you do or do not meet the preconditions of the Activity to which you wish to transition. The choice of routes reflects the opportunistic, event-based process modeling approach embodied in OPEN—in some senses more realistically described as a metaprocess model.

This architecture also illustrates an instantiation of the proposed Architecture Reference Model[10] that organizes information into four main categories:

- Business Strategy
- Business Process
- Business Components
- Infrastructure

The Business Strategy issues, or the statements the business makes about its goals and objectives, form the backdrop to the Business Processes that implement the strategies in the form of workflows—seen here in the Activities of Program Planning and Domain Modeling. The Business Components of the Architecture Reference Model are also represented in the Domain Modeling Activity and in the Analysis and Model Refinement Activities (supported by the group of BPR-related Techniques, for instance) and, finally, the Technical Infrastructure is mirrored in the life cycle model by the Build Activities.

On the left hand side of Figure 2 are Activities that are associated with a single project (discussed here); on the right hand side, in the shaded box, are those Activities that transcend a single project and are associated more with strategic planning and implementation concerns, e.g., resources across several projects; reuse strategies; delivery and maintenance aspects (not discussed here). OPEN includes both projects and organizational software strategies.

OPEN TASKS OPEN Tasks can be grouped. Some occur typically earlier in the life cycle; others group around a particular domain such as distribution or database management. Details on OPEN Tasks (summarized here) can be found in Henderson-Sellers *et al.*[7]

Tasks that focus on user interactions and business issues relate to:

- problem definition and user requirements
- business process engineering
- approval to proceed
- business object modeling

Tasks that focus on large scale architectural issues include:

- architecture
- frameworks
- optimization

Tasks that focus on project management issues, which are described in detail in Henderson-Sellers and Due,[11] include:

- development of software development context plans and strategies
- development and implementation of resource allocation plan
- a feasibility study

Tasks that focus on reuse issues include:

- optimize reuse (with reuse)
- create new reusable components
- manage library of reusable components

There are also groups of tasks devoted to database issues, distributed computer systems, and modeling/building the system.

OPEN TECHNIQUES Tasks are accomplished by Techniques. Techniques are ways of doing things. They include the ways that have been tried and tested over the last decade; but also may include new techniques that are more experimental. Some indication of the level of maturity of the individual technique is thus given as part of its full specification.

Most of the world's leading methodologists have been actively pursuing method unification.

One or more techniques are used to accomplish a Task. The developer chooses the appropriate Task(s) from those described in OPEN or from their own experience, sometimes choosing between two competing alternatives. Thus, for example, in order to find objects, the choices may be, say, using use cases, using textual (noun) analysis, identifying concepts and their responsibilities, using CRC cards, etc. In reality, many tasks are best accomplished by a combination of techniques rather than just one. There are too many cases of the use of a single technique being taken to an extreme. For example, at one conference, a story of noun analysis being used

continued on page 55

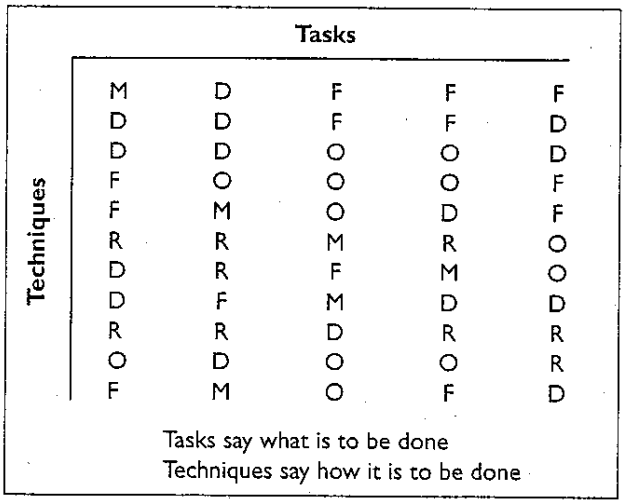


Figure 3. The heart of OPEN is a two-dimensional relationship between Tasks and Techniques Activities. At any life cycle stage, many tasks may be being undertaken; and for each task, a number of techniques may be useful. For each combination of task and technique, an assessment can be made of the likelihood of the occurrence of that combination. Some combinations can be identified as mandatory (M), others as recommended (R), some as being optional (O), some are discouraged (D) but may be used with care and other combinations that are strictly verboten (F = forbidden).[7]

7. Tsichritzis, D.C. and F.H. Lochovsky. *Data Models*, Prentice-Hall, Upper Saddle River, NJ, 1982.
8. Atwood, T., *et al. The Object Database Standard: ODMG-93, Release 1.2*, Cattel, R.G.G. (ed.), Morgan Kaufmann, San Mateo, CA, 1996.
9. Kim, W., E. Bertino, and J.F. Garza. "Composite Objects Revisited," *Proceedings of the 1989 ACM SIGMOD*, SIGMOD RECORD, 18(2):337–347, 1989.
10. Ehlmann, B.K., G.A. Riccardi, and L.C. Dennis. "Representing non-inheritance relationships in an object-oriented, scientific database," *Proceedings of the Sixth International Working Conference on Scientific and Statistical Database Management*, ETH Zurith, June 1992, pp. 99–109.
11. Ehlmann, B.K., L.C. Dennis, and G.A. Riccardi. "An object-based conceptual model of a nuclear physics experiments database," *Nuclear Instruments & Methods in Physics Research, Sect. A*, Elsevier Science/North-Holland, New York, 1993, pp. 294–308.
12. Ehlmann, B.K. and G.A. Riccardi. "A notation for describing aggregate relationships in an object-oriented data model," *Applications of Databases—first International Conference Proceedings*, Litwin, W. and T. Risch (Eds.), Lecture Notes in Computer Science 819, Springer-Verlag, 1994, New York, pp.62–77.
13. Ehlmann, B.K. and G.A. Riccardi. "A comparison of ORN to other declarative schemes for specifying relationship semantics," *Information and Software Technology*, Elsevier Science, 38(7):455–465, July 1996.
14. Technical Committee X3H2—Database. *Database Language SQL*, American National Standards Institute, 1992.
15. Hardeman, S.K. and B.K. Ehlmann. "Relationship behavior in object databases: Subtleties and inconsistencies," *Proceedings 34th ACM Southeast Conference*, 1996, pp. 224–229.
16. Object Design Inc.,*ObjectStore Technical Overview—Release 3*, Object Design, Burlington, MA, March, 1994.

Methods Unification

continued from page 43

for a 300-page requirements specification rightly created disbelief in the audience at such a gross misapplication of the technique.

Techniques are intrinsically orthogonal to the notion of Tasks. They cannot readily be grouped in any unique way. They are akin to the tools of the tradesperson—a carpenter's toolbox contains many tools, some of which have superficial resemblances but may have operational affinity to tools of different outward appearance. Because a full description of the OPEN toolbox of Techniques (more than 150) is a book in itself, we will not discuss them in this column.

OPEN PRINCIPLES Finally, OPEN embodies a set of (object-oriented) principles. It permits enhanced semantics for object models based on the contributions of methods such as SOMA, BON, Syntropy, Firesmith, etc. Furthermore, OPEN is fully object-oriented in that encapsulation is a basic principle. To this end, bi-directional associations are not first-order members of the metamodel, but are instead modeled as a pair of unidirectional associations that are semi-strong inverses of each other. It is a logical consequence of this that class invariants are not an optional extra in the modeling semantics.[12] Rulesets, which generalize class invariants, can be used to model intelligent agents as objects.

In OPEN, OO principles are basic and should be adhered to. These include:

- object modeling as a very general technique for knowledge representation
- encapsulation
- polymorphism

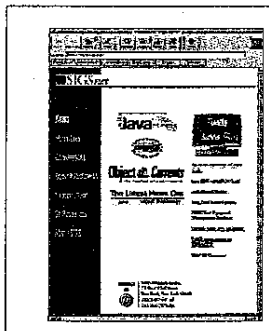
together with

- clear, jargon-free, and well-founded definitions of all terms
- extensive use of abstraction techniques, a foundation for semantically cohesive and encapsulated "objects"

CONCLUSIONS OPEN is a fully object-oriented life cycle methodology. It unifies MOSES, SOMA, and Firesmith and utilizes tips and techniques from a large number of other extant OO methods. The heart of OPEN is a fully object-oriented life cycle model in which Activity objects have Tasks as operations, together with a two-dimensional matrix linking Tasks with Techniques. This gives tailorability and full support for both technical and management issues in a wide number of technical domains. ■

References

1. Henderson-Sellers, B., and D. Firesmith. "COMMA: Proposed core model," *Journal of Object Oriented Programming (ROAD)* 48–53, Jan. 1997.
2. Anon. OMG's technical committee meets on object analysis and design proposals, *Object Magazine* (March 1997, in press).
3. Henderson-Sellers, B., D. Firesmith, and I. Graham. "OML metamodel: Relationships and state modelling," *Journal of Object Oriented Programming (ROAD)*, 10(1):47–51, Mar./Apr. 1997.
4. Firesmith, D., B. Henderson-Sellers, and I. Graham. *OPEN Modeling Language (OML) Reference Manual*, SIGS Books, New York, 1997.
5. Henderson-Sellers, B. "Who needs an OO methodology anyway?," *Journal of Object Oriented Programming*, 8(6):6–8, 1995.
6. Henderson-Sellers, B., *et al.* "OPEN: Toward method convergence?" *IEEE Computer*, 29(4):86–89, 1996.
7. Henderson-Sellers, B., *et al. The OPEN Heart, TOOLS 21*, C. Mingins, R. Duke and B. Meyer, Eds., TOOLS/ISE, 1996.
8. Graham, I. M. *Migrating to Object Technology*, Addison-Wesley, Wokingham, 1995.
9. Graham, I. M. "A non-procedural process model for object-oriented software development," *Report on Object Analysis and Design*, 1(5):10–11, 1995.
10. Hertha, W. "The architecture reference model," position paper to the Business Object Design and Implementation Workshop, OOPSLA '95, Austin, TX, Oct. 1995 (available from http://www.tiac.net/users/oopsla/hertha.html).
11. Henderson-Sellers, B. and R.T. Due. "OPEN project management," *Object Expert*, 2(2):30–35, 1997.
12. Graham, I.M., J. Bischof, and B. Henderson-Sellers. "Associations considered a bad thing," *Journal of Object Oriented Programming*, 9(9): 41–48, Feb 1997.



http://www.sigs.com

JOURNAL OF OBJECT-ORIENTED Programming

May 1997
Vol. 10, No. 2

Cover image: David Bishop