Manuscript: ROAD966.TEX

Methods Unification: the OPEN methodology

B. Henderson-Sellers

School of Computer Science and Software Engineering
Swinburne University of Technology, Hawthorn, Victoria, Australia

I.M. Graham

Ian Graham and Associates, London

D. Firesmith

Knowledge Systems Corporation, Cary, NC, USA

Printed: September 28, 1998

[*JOOP (Report on Object Analysis and Design)* (June 1997)]

Address for Correspondence

Professor B. Henderson-Sellers
Director, Centre for Object Technology Applications and Research
School of Computer Science and Software Engineering
Swinburne University of Technology
John Street
PO Box 218
Hawthorn
Victoria 3122
AUSTRALIA
fax: +61 3 9819 0823
email: brian@csse.swin.edu.au

Methods Unification

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. Whilst competition may be fun, it soon becomes clear that to those of us who really believe that OT 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 which I have been reporting in this JOOP/ROAD column over 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 whilst permitting extensions *beyond* the core so that each methodology can be directed at its own market.

Whilst the last point is still to be fully realized, there are two unification projects which have influenced and been influenced by all this effort on metamodelling. In the first, Grady Booch and Jim Rumbaugh collaborated to create the Unified Method, Version 0.8

which later became the Unified Modeling Language, Version 0.9 (and onwards) or UML. UML also involves contributions from Ivar Jacobson.

UML consists of a notation and an underlying metamodel.

The second unification effort started by a merger announced 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 at a stroke three other methodologies. In addition to these three, there are 25 other OO proponents in the OPEN Consortium dedicated to producing the OPEN methodology.

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

The heart of OPEN

OPEN is a full lifecycle, process-focussed methodology which supports all the elements of a methodology that one would expect[5]:

- a collection of rules and guidelines
- 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 lifecycle i.e. process
- identification of organizational rôles e.g. business analyst, programmer

- 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 lifecycle Activities represented as objects (Figure 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 which 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 fulfil the Tasks. This is accomplished by the use of a matrix (Figure 3) which 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 which 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). Since we are modelling these activities as objects, we can associate contracts with each Activity object (hence the lifecycle name of "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 lifecycle 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 this figure 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 pre-conditions of the Activity to which you wish to transition. The choice of routes reflects the opportunistic, event-based process modelling approach embodied in OPEN — in some senses more realistically described as a meta-process model.

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

- Business Strategy
- Business Process
- Business Components
- Infrastructure

The Business Strategy issues, the statements the business makes about its goals and objectives, form the backdrop to the Business Processes which implement the strategies in

the form of workflows — seen here in the Activities of Programme Planning and Domain Modelling. The Business Components of the Architecture Reference Model are also represented in the Domain Modelling 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 lifecycle model by the Build Activities.

On the left hand side of Figure 2 are Activities which are associated with a single project (discussed here); on the right hand side, in the shaded box, are those Activities which 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 lifecycle; others group around a particular domain such as distribution or database management. Details on OPEN Tasks (summarized below) can be found in ref. 7.

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

- Problem definition and user requirements
- Business process engineering
- Approval to proceed
- Business object modelling

Tasks which focus on large scale architectural issues include:

- Architecture

- Frameworks
- Optimization

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

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

Tasks which 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 modelling/building the system.

OPEN Techniques

Tasks are accomplished by Techniques. Techniques are thus 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 on the level of maturity of the individual technique is thus given as part of its full specification.

One or more techniques is used to accomplish a Task. The developer chooses the appropriate Task(s) from those described in OPEN or from their own experience, sometimes selecting between two competing alternatives. Thus, for example, in order to find objects, the choice may be between, 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 mixture 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 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 resemblences but may have operational affinity to tools of different outward appearance. Since a full description of the OPEN toolbox of Techniques (over 150) is a book in itself, we cannot 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 modelled as a pair of uni-directional 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 modelling 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 modelling 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 lifecycle 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 lifecycle 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 Firesmith, D., 1997, COMMA: proposed core model, *J. Obj.-Oriented Prog. (ROAD)*, January 1997

[2.] Anon, 1997, OMG's technical committee meets on object analysis and design proposals, *Object Magazine* (March 1997, in press)

[3.] Henderson-Sellers, B., Firesmith, D. and Graham, I., 1997, COMMA: its influence on OPEN, *J. Obj.-Oriented Prog. (ROAD)*, March/April 1997

[4.] Firesmith, D., Henderson-Sellers, B. and Graham, I., 1997, *OPEN Modeling Language (OML) Reference Manual*, SIGS Books, NY

[5.] Henderson-Sellers, B., 1995, Who needs an OO methodology anyway?, guest editorial for *J. Obj.-Oriented Programming,* **8(6)**, 6–8

[6.] Henderson-Sellers, B. and Graham, I.M. with additional input from C. Atkinson, J. Bézivin, L.L. Constantine, R. Dué, 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

[7.] Henderson-Sellers, B., Graham, I.M., Firesmith, D., Reenskaug, T., Swatman, P. and Winder, R., 1996, The OPEN heart, *TOOLS 21*, (eds. C. Mingins, R. Duke and B. Meyer), TOOLS/ISE, 187–196

[8.] Graham, I.M., 1995, *Migrating to Object Technology*, Addison–Wesley, Wokingham

[9.] Graham, I.M., 1995, A non-procedural process model for object-oriented software development, *Report on Object Analysis and Design,* **1(5)**, 10–11

[10.] Hertha, W., 1995, The architecture reference model, position paper to the Business Object Design and Implementation Workshop, OOPSLA '95, Austin, TX, October 1995 (available from http://www.tiac.net/users/oopsla/hertha.html)

[11.] Henderson-Sellers, B. and Dué, R.T., 1997, OPEN project management, invited article for *Object Expert,* **2(2)**, 30–35

[12.] Graham, I.M., Bischof, J. and Henderson-Sellers, B., 1997, Associations considered a bad thing, *J. Obj.-Oriented Programming* (Feb 1997)

Figure Legends

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

Figure 2 Contract-driven process lifecycle model (adapted from ref. 6)

Figure 3 The heart of OPEN is a two-dimensional relationship between Tasks and lifecycle Activities. At any lifecycle stage, many tasks may be being undertaken; and for each task, a number of techniques may be useful. For each combination of lifecycle activity and task, 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) (after ref. 7).