OPEN: Towards Method Convergence

B. Henderson-Sellers and I.M. Graham

with other input from the OPEN methodology collaborative team of
C. Atkinson, J. Bézivin, L.L. Constantine, P. Desfray, R. Dué,
R. Duke, D. Firesmith, G. Low, J. McKim,
D. Mehandjiska-Stavrova, B. Meyer, J.-M. Nerson, J.J. Odell,
M. Page-Jones, T. Reenskaug, B. Selic, A.J.H. Simons, P. Swatman, R. Winder

This paper was published in *IEEE Computer*, Volume 29 number 4, IEEE Computer Society, Los Alamitos, CA, USA, 86-89

In the beginning, there were no object-oriented (OO) methodologies. Around 1990, a collection of analysis and (predominantly) design techniques were published, either in paper or book form. These approaches were, in general, derived independently and (possibly) tested with local industry. Over the next few years, the increasing number of publications and international conferences allowed the OO methodologists to meet, discuss and exchange ideas. This then led to a "second round" of publications, which may loosely be termed second generation in the sense that they unashamedly borrow from other sources. Examples are Fusion (using CRC, Booch, OMT and formal methods) and the second edition of Booch's "Object-Oriented Analysis and Design". These second generation approaches do not invalidate the first generation approaches (unless they are a second version e.g. Booch93/94 *does* replace the 1991 version). Consequently, by the mid-1990s methods such as RDD, OMT and OOSE co-existed with more recently published methodologies such as Booch93/94, MOSES, SOMA, Martin/Odell, BON and OOram.

Whilst Grady Booch and Jim Rumbaugh continue to focus on OO modelling techniques as they develop "a method for specifying, visualizing and documenting the artifacts of an object-oriented system development", there is no stated intention to include in their joint work any process model or any significant concern or focus on business issues, strategic planning, project management, reuse strategies, requirements engineering, metrics, legacy systems, interface design, transaction processing or formal specifications. The stated aim of the OMEGA project[1], creating the first, fully-tested third generation methodology, OPEN (Object-oriented Process, Environment and Notation), is to do just this: provide a FULL process model underpinning for systems to be developed, initially within the business context, across the full lifecycle including reuse, quality, transactions and legacy systems. In addition, OPEN will offer support for HCI, concurrency, databases, distributed systems, fuzzy logic, AI and intelligent agents. Incorporating, as it does, many ideas from methodologies such as RDD, BON, MOSES, SOMA, Martin/Odell, Syntropy, ROOM, OBA, OOram, FOOM etc., OPEN is a truly third generation methodology. In addition, many of those authors have pledged active support of OPEN, as evidenced by the author list for this column.

The good news, therefore, is that OPEN is *not* just another methodology. We do not add one more to the overall count of current OO methodologies. On the contrary, the OPEN methodological framework outlined here will significantly decrease the number of methods available to be chosen from, superseding as it does MOSES, SOMA, BON, Martin/Odell, etc. The draft handbook for the OPEN methodology is currently in the hands of the methodology developers and the industry users — names such as Dow Jones Telerate (in Sydney), SBC Warburg (in London), JP Morgan, FourFront (in India), LBMS. Only following industry testing will we publish the definitive OPEN Handbook.

The overall OPEN life-cycle process model

OPEN is fully object-oriented. Even its life-cycle model is represented as a network of interacting objects (Figure 1(a)). Each object represents an Activity. Transition between Activities is governed by a contract (usually pre- and post-conditions and invariants) — we thus call this life-cycle the contract-driven life-cycle. Activities include a well-specified testing programme and should deliver test results against both task scripts and a normal technical test plan.

The contract-driven lifecycle *does not* prescribe only one deterministic process. Rather, it permits (1) ordering in an iterative and incremental fashion and (2) the tailoring by a given organization to its own internal standards and culture — an important component for each organization's internal "culture". We indicate in this diagram only *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 post-conditions are your assurance of quality. Different process configurations, chosen to reflect the needs of different problems and different organizations, can be constructed from this flexible framework. Once chosen, the lifecycle process is fixed — although still, at least in an OO project, highly iterative, flexible and with a high degree of seamlessness[2].

On the left hand side of Figure 1(a) are Activities which are associated with a single project; 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. OPEN includes both projects and organizational software strategies. The relation of the process lifecycle in Figure 1(a) to the business issues is illustrated in Figure 1(b) in which it can be seen that software development can be viewed as a recurring set of enhancements following the first of the iterative cycles (i.e. the growth period).

Activities and Tasks

The Activities permit a largescale structuring of the project management for an OO product development. What they don't do is to identify things that have to be done at a low enough resolution to match with tasks that need to be done in order to meet the objectives of each Activity. For example, when the developer is focussing on an Activity such as Evolutionary Development, there are a number of associated Tasks which need to be undertaken successfully. One of these is the Task: Structure the object model. This is a clearly delineated task that someone can take responsibility for (the delivery of a well-structured object model for the problem in hand). We should also note that for this particular Activity, more than one task may well be identified; for other Activities there may be predominantly only one Task required. OPEN's Tasks are therefore statements of things that need to be done. They may be described as the "smallest unit of work on a schedule which contributes to a milestone". Tasks are either completed or not completed.

We can thus think of these links (Activities to Tasks) in a probabilistic manner and visualize them in a 2-D matrix which links together each Activity with each Task and allocates to that linkage a measure of possibility of its occurrence. Following industry tests, we will give our overall recommendations for each Activity/Task pair in terms of M (=mandatory), R (recommended) O (=optional), P (=possible — but unlikely) and F (=forbidden). For example, planning the number of iterations in an OO development is not likely to occur in the Evaluation Activity. Finding the CIRTs (CIRT is a generic term to include class, instance, rôle and type) is almost certainly mandatory in the Development Activity (Rapid OOA/D/P and testing). Constructing a Distributed Systems Strategy may be totally absent in some software projects; but critical to the project's success in others. For any specific project the actual pattern of M/R/O/P/F should be determined as part of the initial planning activity, usually the responsibility of the project manager. This is part of the process of tailoring the OPEN methodology to your own organization.

This two-dimensional matrix also offers the project manager significant flexibility. If new ideas/tasks are developed in a particular context, then incorporating them into this framework is extremely easy. It only requires the addition of a single line in the matrix and the identification of the M/R/O/P/F nature of the interaction between this new task and the activities of the chosen lifecycle process model.

Tasks and Techniques

However, knowing that the Task: Structure the object model is one (of several) tasks that the developer can undertake in order to complete the Activity does not tell the developer *how* to accomplish the Task(s). The Task is the statement of the unit of work required, the "what"; the "how" is described by one or more Techniques. This is similar to saying that I want to hang a picture on my living room wall. This is today's Task. But how do I do that? I use some Technique — but often I have a choice between techniques; and furthermore I may use not just one but several techniques to accomplish my task. In this case I have a choice between using (a) a hammer to knock a nail into the wall or (b) a screwdriver and a screw *together with* (for either option) a piece of string and knowledge of how to tie an appropriate (reef) knot.

OPEN does not *mandate* techniques — you are at liberty to accomplish these tasks with whatever tools and techniques you are familiar with. However, in order to aid the developer, the OPEN methodology does include a large range of *suggested* techniques which we, and others, have found to be appropriate. Again there is a fuzzy nature to the linkage of techniques and tasks. Some tasks are clearly best accomplished with a single, specific technique — a technique applicable to that task and nothing else (for example, implementation of services which support the coding task). Other techniques will be found useful in a range of tasks (for example, contract specification). And finally for some tasks there may be a choice that the project manager has to make. For example, there are many

ways of identifying classes, objects and types. These include interactive techniques such as the use of CRC-like cards to identify responsibilities; scenarios/task models/scripts/use cases to focus on functionality delivered as a prelude to finding CIRTs within these scripts; textual analysis, in which nouns in the requirements analysis have the potential to be realized as CIRTs; simulation which focusses on the objects within the modelling exercise; and even (for some skilled people), the use of ER diagrams as a basis for an CIRT structure diagram.

OPEN has over 100 suggested and proven techniques from which to choose. Many focus on business issues (e.g. cost estimation and project planning), others on reuse (completion of abstractions, generalization, patterns and frameworks). We believe in the importance of quality, through techniques such as contracting[3]. User requirements capture is also fully supported by the use of task scripts, a more generic version of use cases[4] which provide a seamless link between business objects and (software) system objects. Tasks also provide the basis for the task point metric — a focus of the International OO Metrics Club.

The OPEN technology supports a metamodel compatible with the emerging OMG standard, a fully OO, responsibility-driven and contract-driven object model with connections which do not break encapsulation, a suite of techniques to manage large models and their inherent complexity, technical support for rôles, patterns, frameworks and distributed object systems — all supported by a full notation and a growing number of CASE, requirements engineering, metrics and BPR tools such as SOMATiK, Simply Objects, MetaEdit, Graphical Designer.

Summary

OPEN is a methodology that contains many options from which to choose so that when that choice has been made the resulting method, tailored to your organization and your project, is readily learnable and becomes your organizational standard; yet "your own methodology" is completely compatible with the published (and larger, more comprehensive) OPEN methodology, supported worldwide by consulting and training organizations (e.g. Tower Technology, Vayda Consulting Inc., COTAR), CASE tool vendors (e.g. Bezant Object Technologies, Advanced Software Technologies, Inc., Adaptive Arts Pty Ltd, Taskon) and users (e.g. Dow Jones Telerate, SBC Warburg) — a methodology which opens the door to your software development future.

References

[1] Henderson-Sellers, B., 1995 Methodology — convergence is in the air!, *Object Magazine,* **5(7)**, 61

[2] Waldén, K. and Nerson, J.-M., 1995 *Seamless Object-Oriented Architecture*, Prentice Hall, 301pp

[3] McKim, J., 1996, Programming by contract, *IEEE Computer*, March 1996

[4] Graham, I.M., 1996, Task scripts, use cases and scenarios in object-oriented analysis, *Object-Oriented Systems,* **3(3)**, 123-142

Methodologies consulted: (perhaps as a side bar)

Booch93/94 — Booch, G., 1994, *Object-Oriented Analysis and Design with Applications* (2nd edition), The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 589pp

BON: Business Object Notation — Waldén, K. and Nerson, J.-M. 1995, *Seamless Object-Oriented Architecture*, Prentice Hall, 301pp

CRC: Class, Responsibility, and Collaborations — Beck, K., and Cunningham, W., 1989, A laboratory for teaching object-oriented thinking, *SIGPLAN Notices*, **24(10)**

FOOM: Formal Object-Oriented Methodology — Swatman, P., 1995, Documentation from http://mae.ba.swin.edu.au/∼paul/foom.html

Fusion — Coleman, D., Arnold, P., Bodoff, S.., Dollin, C., Gilchrist, H., Hayes, F. and Jeremaes, P., 1994, *Object-Oriented Development: the Fusion Method*, Prentice Hall Inc., Englewood Cliffs, NJ, 313pp

Martin/Odell — Martin, J. and Odell, J.J., 1995, *Object-Oriented Methods. A Foundation*, PTR Prentice Hall, New Jersey, 412pp

MOSES: Methodology for Object-oriented Software Engineering of Systems — Henderson-Sellers, B. and Edwards, J.M., 1994a, *BOOKTWO of Object-Oriented Knowledge: The Working Object*, Prentice Hall, Sydney, 616pp

OMT: Object Modeling Technique — Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., 1991, *Object-oriented Modelling and Design*, Prentice Hall, New Jersey, 500pp

OOram: Object-Oriented Role Analysis Method — Reenskaug, T., Wold, P. and Lehne, O.A., 1996, *Working with Objects. The OOram Software Engineering Manual*, Manning, Greenwich, CT, USA, 366pp

OOSE: Object-Oriented Software Engineering — Jacobson, I., Christerson, M., Jonsson, P. and Övergaard, G., 1992, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, 524pp

RDD: Responsibility Driven Design — Wirfs-Brock, R.J., Wilkerson, B. and Wiener, L., 1990, *Designing Object-Oriented Software*, Prentice Hall, 368pp

ROOM: Realtime Object-Oriented Method — Selic, B., Gullekson, G. and Ward, P.T., 1995, *Real-Time Object-Oriented Modelling*, John Wiley & Sons, Inc., New York, 525pp

Syntropy — Cook, S. and Daniels, J., 1994, *Designing Object Systems*, Prentice Hall, UK, 389pp

SOMA: Semantic Object Modelling Approach — Graham, I.M., 1995, *Migrating to Object Technology*, Addison–Wesley, Wokingham, UK, 552pp

Figure Legends

Figure 1(a) Contract-driven process lifecycle model

Figure 1(b) The process lifecycle is embedded in an iterative cycle of growth and enhancements