Using OPEN's Deontic Matrices for E-Business

B. Henderson-Sellers, D. Lowe and B. Haire
University of Technology, Sydney

Key words : Project and process management - E-Business projects

1. INTRODUCTION

Method engineering (e.g. Brinkkemper *et al.*, 1998; Ralyté and Rolland, 2001) focuses on how to create a method (here a software development method) that is highly suitable and “personalized” for a particular project or organization. Successful method engineering needs, as input, a repository of process elements (Chroust, 2000) that describes individual elements of the methodological approach; these include descriptions of activities, phases, tasks, techniques, roles, deliverables and so on. Given such a repository, sometimes called a process reference model (Henderson-Sellers *et al.*, 2002), a number of various kinds of method or methodology can be constructed (Rupprecht *et al.*, 2000; Henderson-Sellers, 2002a,b).

In this paper, we examine how the use of a deontic¹ matrix, originally introduced by Henderson-Sellers and Edwards (1994) and Graham (1995a) provides possibility values for identifying useful process components and how they should be interconnected. We use industry data for one particular business domain, that of e-commerce, to realize the deontic matrices as used in the OPEN Process Framework (Firesmith and Henderson-Sellers, 2002). The software development paradigm used will be that of OO (object-oriented) and

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35614-3_21](https://doi.org/10.1007/978-0-387-35614-3_21)

C. Rolland et al. (eds.), *Engineering Information Systems in the Internet Context*

© IFIP International Federation for Information Processing 2002

CBD (component-based development). Values are given for both large and small B2C examples.

Before progressing further, two points are important to note. The first is that the nature of Web development processes is currently rather poorly addressed within the research literature (see, for example, Lowe and Henderson-Sellers (2001a; 2001b)). The purpose of the research described in this paper is to begin to understand and structure our understanding of how to effectively construct Web development processes. In this context, we have adopted a grounded theory research approach –progressively collecting and analysing data on specific ‘experimental situations’ (i.e. actual commercial projects) within the context of an emerging theoretical framework.

Secondly, although the discussion in this paper focusses on applying concepts within the framework provided by OPEN, the general lessons with regard to selection of process components is much more broadly applicable. For example, even when OPEN is not being explicitly utilised, the deontic matrices provide guidance on the relevance of various activities and tasks within Web projects.

2. BACKGROUND: THE OPEN PROCESS FRAMEWORK

The OPEN Process Framework (OPF) contains not only a repository of process elements but also an underpinning metamodel. This second element is seen to be vital in creating a fully flexible framework in which true method engineering becomes possible.

The OPEN Process Framework is one example of a metamodel-based set of process components collected in the OPF Repository (Figure 1). The OPF itself has three major elements, as seen in this figure: the metamodel, the repository of process components, each of which is an instance of an element in the metamodel, and the process construction and tailoring guidelines. These latter guidelines are used not only in constructing and expanding the repository of process elements, as seen for example, in Haire *et al.* (2001), but also by the process engineering in constructing a personalized OO development process for use in a specific organization or on a specific project.

Figure 1 : Creating a personalized OO development process from the OPF (using UML as the notation)

The elements of the OPF metamodel provide a solid basis for generation of the process components and providing the rules by which each of the elements (process components) can be inter-related. There are five major groups of meta-elements defined in Firesmith and Henderson-Sellers (2002). These are shown in Figure 2.

The central triad is that of Producer, Work Unit and Work Product. Producers are people, teams and tools playing Roles in which they undertake Work Units (i.e. actions such as Activities and Tasks) in order to create and maintain a pre-specified number of Work Products (a.k.a. deliverables). Work products include graphical and textual descriptions, the former of which, in an OO/CBD world, tend to be documented using a modelling language such as the UML (OMG, 2001).

Figure 2 : The basis metatypes of the OPEN Process Framework (OPF) (after Firesmith and Henderson-Sellers, 2002)

As well as the three meta-elements discussed above, there also, in the metamodel, classes to represent Stages (phases, cycles etc.) and Languages (natural languages, programming languages, modelling languages). Stages provide the overall macro-scale, temporal organization to the process; Languages are used mainly in documentation (of the Work Products).

3. LINKING ACTIVITIES, TASKS AND TECHNIQUES – THE DEONTIC MATRICES

As noted earlier, there are several kinds of guidelines needed to engineer a project-specific process. These include, *inter alia*, process construction guidelines, tailoring guidelines and, less relevant here, extension guidelines – which assist the process engineer in modifying the metamodel itself. (Tailoring guidelines, which support minor modifications to the process once constructed also have less immediate impact on the topic of this paper.) Here we focus on one element of the process construction guidelines (viz. the use of deontic

matrices – see below) and provide some empirical evidence supporting its use. Other important elements (not discussed further here) include sequencing rules, which can be expressed using pre- and post-conditions on (particularly) Tasks (e.g. Graham, 1995b) and/or by ensuring the process and product perspectives are adequately connected (Brinkkemper *et al.*, 1998).

A *process construction guideline* helps process engineers both to instantiate (when necessary) the development process framework (metamodel) to create process components and also to select the best process components (from the Repository) in order to create the process itself (e.g. Brinkkemper *et al.*, 1998; Firesmith and Henderson-Sellers, 2002). Specifically, guidance is provided on the selection of appropriate work products, producers and work units as well as advising on how to allocate tasks and associated techniques to producers and how to group the tasks into workflows, activities etc. Finally, developmental stages (including phases and lifecycles) are chosen.

Figure 3 : Various kinds of Work Unit depicted using UML as the notation

In this paper, we focus on the selection of, and more importantly the connexions between, various instances of the Work Unit metaclass. The prime subtypes (in the metamodel) are Activity, Task and Technique (Figure 3)

Activities, which may also be made up recursively of subActivities, contain Tasks in a many-to-many relationship. To describe this multifaceted connexion, the OPF suggests the use of a deontic matrix which specifies the possibility value for each pair of Activity and Task instances (i.e. each process component derived from the Activity and Task metaclasses respectively). Figure 4 shows, schematically, an example of how this might look.

Figure 4 : Schematic deontic matrix linking OPEN's Activities (the columns) to Tasks (the rows)

Possibility values are now entered into this matrix, for each Activity/Task pairing, *for a specifically constructed (or to-be-constructed) and configured process instance*. In the OPEN approach (Graham *et al.*, 1997; Henderson-Sellers *et al.*, 1998; Henderson-Sellers and Unhelkar, 2000) these values are recommended to be selected from one of five values: M = mandatory; R = recommended; O = optional; D = discouraged; and F = forbidden². The inclusion of optionality in the choice of Tasks and Techniques permits an organization or a project manager to select tasks and techniques specially suited to local conditions – including skills sets of project members, resource availability, level of criticality/safety requirements and so on. We do anticipate, however, that for most organizations these specific characteristics will remain static for the duration of the project (and often longer) so that the majority of the values in the

deontic matrix are likely (our conjecture only) to tend towards becoming bimodal i.e. using only M and F values.

Having created a deontic matrix to link Activities and Tasks together, we then examine the other many-to-many linkage in Figure 3 – between Task and Technique. For each selected task in OO/CBD there are frequently options for the most appropriate technique. Factors affecting optimization of the choice of technique include skills sets, CMM level, tool availability and so on. What matters is that the Task is successfully completed, *not* that any particular technique is used to effect the Task. Consequently, methodologists find it impossible to prescribe, or hardcode into their methodologies, specific techniques³. For the user (via the assistance of the process engineer) to be sure of the suite of techniques most suitable to their organizational context, a second deontic matrix is introduced which, similarly to Figure 4, links Tasks (the columns) to Techniques (the rows).

In this paper, we do not attempt here to construct these two deontic matrices for a specific organization or a specific project but rather to offer guidelines and “rules of thumb” for specific *categories* of web developments; notably small/medium business-to-customer (B2C) and large business-to-customer ecommerce situations. The data used to construct the values in these matrices (in the next section) were derived from case studies of two different organisations in differing industries, though both developing e-commerce applications. Thus, we can propose that the values in the matrices will be generically applicable to that domain – though possibly that domain only. To complete the values in the two deontic matrices for other domains and technology situations, additional empirical data must be sought.

4. REALIZING THE DEONTIC MATRICES OF OPEN

The number of activities that are required in the personalized process/method can be roughly equated to the size of the project. In other words, as a project increases in scale the complexity of the process increases and hence we introduce additional process activities. For example, one would expect that a small project would not contain activities such as *Resource Planning*; whereas large projects would

include a more complete set of the activities selected from those within the OPF Repository. The number of tasks may well also relate to the general project “size” in the sense that a team undertaking a larger project are more likely to be more experienced and thus have more sophisticated skills at their disposal. Indeed, different members of a highly skilled team may well use different techniques to accomplish the same task. This leniency is, however, highly dangerous for the inexperienced developer, when a more strict technique selection should be made and the use of those selected techniques mandated and monitored. In other words, many OO tasks and techniques require experience before they can be recommended for practical use whereas others can be confidently enacted by relative OO novices. Such considerations remain, however, outside the scope of this paper, in which we will take a much broader “average” viewpoint on the B2C e-business category. The novelty of the results is seen in the Activity/Task linkages whereas, interestingly, our empirical evidence suggests that the Task/Technique linkage for B2C exhibits little significant difference from regular (i.e. non ecommerce) applications development and so will not be discussed further herein.

4.1 Small/Medium Business-to-Customer (B2C)

Within most (if not all) Business-to-Customer (B2C) web projects, there is a significant emphasis placed on the market research aspect of the requirements engineering, although small to medium projects tend to have less of an in-depth analysis and design phase than do large projects. This does add to the overhead of the project, which thus constitutes a large percentage of the overall cost if the project budget is small. This is reflected in the deontic matrix between Activities and Tasks. The tables associating activities to tasks for small/medium business-to-customer projects can be found in Figure 5. As noted earlier, these values were derived from two in-depth case studies at commercial sites in Australia in late 2000 (Haire, 2000). We can see that small systems tend not to use distribution, that architectural design is recommended but not mandatory, that building a white site is seen as optional and that component selection tasks are minimal. Reuse is also typically eschewed; although testing and UI design are seen to be of high priority. Resource planning tends also to be typically seen as of little importance.

Tasks	Activities														
	A 1	A 2	A 3	A 4	A 5	A 6	A 7	A 8	A 9	A 10	A 11	A 12	A 13	A 14	A 15
Task 1	O														
Task 2		O	R												
Task 3		R													
Task 4															
Task 5		O													
Task 6		O	O												
Task 7															
Task 8		M													
Task 9		O													
Task 10															
Task 11		O													
Task 12			M												
Task 13			O										O		
Task 14			O												
Task 15			R												
Task 16			O												
Task 17			O												
Task 18															
Task 19			O												
Task 20				R											
Task 21				O											
Task 22					D										
Task 23					D										
Task 24					D										
Task 25					O										
Task 26					O										
Task 27					O										
Task 28						D									
Task 29						O							O		
Task 30						O									
Task 31						O									
Task 32						O					O			O	
Task 33							O								
Task 34							M								

Tasks	Activities												
Task 35						R							
Task 36						R							
Task 37													
Task 38						O							
Task 39						O							
Task 40						M							
Task 41						R							
Task 42						R							
Task 43						R							
Task 44						R							
Task 45						D							
Task 46						D							
Task 47						D							
Task 48						D							
Task 49						D							
Task 50						R							
Task 51						O							
Task 52						M							
Task 53						M							
Task 54						R							
Task 55						R							
Task 56						R							
Task 57						R							
Task 58						R							
Task 59						M		M					
Task 60						D							
Task 61						M							
Task 62						O							
Task 63							O	O				O	
Task 64								D					
Task 65								D					D
Task 66									M	R			
Task 67									O	R			
Task 68									O	O			
Task 69									O	O			
Task 70									R	O			
Task 71									O	O			

[illegible]

Key to Activities and Tasks :

A1	Project initiation
A2	Requirements Engineering
A3	Analysis and Model Refinement
A4	Project Planning
A5	Component Selection
A6	Build
A7	Build – Evolutionary Development
A8	Build – User Review
A9	Build – Consolidation
A10	Evaluation
A11	Programme Planning
A12	Resource Planning
A13	Domain Modelling
A14	Other Projects
A15	Use of system
Task 1	Obtain business approval
Task 2	Develop BOM
Task 3	Identify context
Task 4	Identify source(s) of requirements
Task 5	Conduct market research
Task 6	Create white site
Task 7	Develop data standard
Task 8	Identify user requirements
Task 9	Define problem and establish mission and objectives
Task 10	Establish user requirements for distributed systems
Task 11	Establish user DB requirements
Task 12	Analyze user requirements
Task 13	Undertake architectural design
Task 14	Choose architectural pattern
Task 15	Design and incorporate content management strategy
Task 16	Design and incorporate personalization strategy
Task 17	Develop layer design
Task 18	Establish distributed systems strategy
Task 19	Select database/storage strategy
Task 20	Develop and implement resource allocation plan
Task 21	Obtain business approval
Task 22	Screen the candidate list of component frameworks

Task 23	Evaluate the potential component frameworks
Task 24	Choose appropriate component frameworks
Task 25	Screen the candidate list of components
Task 26	Evaluate the potential components
Task 27	Choose appropriate components
Task 28	Construct the object model
Task 29	Create and/or identify reusable components
Task 30	Construct frameworks
Task 31	Optimise for reuse
Task 32	Optimize reuse ("with reuse")
Task 33	Prepare content
Task 34	Code
Task 35	Integrate components
Task 36	Design and implement physical database
Task 37	Distribution/replication design
Task 38	Operational and performance design
Task 39	Performance evaluation
Task 40	Design user interface
Task 41	Create the role model
Task 42	Create the task model
Task 43	Create the content model
Task 44	Integrate content with user interface
Task 45	Identify CIRTs (classes, instances, roles and types)
Task 46	Determine initial class list
Task 47	Identify persistent classes
Task 48	Identify roles
Task 49	Refine class list
Task 50	Map logical database schema
Task 51	Map roles on to classes
Task 52	Test
Task 53	Perform acceptance testing
Task 54	Perform class testing
Task 55	Perform package/cluster testing
Task 56	Perform regression testing
Task 57	Performance testing
Task 58	Undertake usability design
Task 59	Evaluate quality
Task 60	Analyze metrics data

Task 61	Evaluate usability
Task 62	Review documentation
Task 63	Optimize the design
Task 64	Maintain trace between requirements and design
Task 65	Write manuals and other documentation
Task 66	Model and re-engineer business process(es)
Task 67	Build context (i.e. business process) model
Task 68	Build Task Object Model
Task 69	Convert Task Object Model to Business Object Model
Task 70	Do user training
Task 71	Prepare Invitation To Tender (ITT)
Task 72	Undertake feasibility study
Task 73	Develop and implement resource allocation plan
Task 74	Choose hardware
Task 75	Choose project team
Task 76	Choose toolset
Task 77	Decompose programmes into projects
Task 78	Develop education and training plan
Task 79	Develop iteration plan
Task 80	Develop timebox plan
Task 81	Identify project roles and responsibilities
Task 82	Manage packages/subsystems
Task 83	Set up metrics collection programme
Task 84	Specify individual goals
Task 85	Specify quality goals
Task 86	User dependencies in the BOM to generate first-cut project plan
Task 87	Develop software development context plans and strategies
Task 88	Develop capacity plan
Task 89	Develop contingency plan
Task 90	Develop security plan
Task 91	Establish change management strategy
Task 92	Establish data take-on strategy
Task 93	Integrate with existing, non-OO systems
Task 94	Tailor the lifecycle process
Task 95	Manage library of reusable components
Task 96	Deliver product to customer
Task 97	Undertake in-process review

Tasks	Activities													
Task 48							O							
Task 49							O							
Task 50							R							
Task 51							O							
Task 52							M							
Task 53							M							
Task 54							R							
Task 55							R							
Task 56							R							
Task 57							M							
Task 58							R							
Task 59							M		M					
Task 60							R							
Task 61							M							
Task 62							O							
Task 63								M	O				O	
Task 64									O					
Task 65									D					O
Task 66										M	R			
Task 67										O	R			
Task 68										O	R			
Task 69										O	R			
Task 70										O	R			
Task 71										O	R			
Task 72	O									O				
Task 73											M			
Task 74											O			
Task 75											O			
Task 76											O			
Task 77											R			
Task 78											R			
Task 79											M			
Task 80											M			
Task 81											M			
Task 82											M			
Task 83											R			
Task 84											O			

Tasks	Activities												
Task 85										R			
Task 86										M			
Task 87										M			
Task 88										R			
Task 89										M			
Task 90										M			
Task 91										M			
Task 92										M			
Task 93										R			
Task 94										M			
Task 95												R	
Task 96													M
Task 97													R
Task 98													R

Figure 6 : Activity Task Matrix for Large B2C.
(For Key to Activities and Tasks see Figure 5)

4.3 Comparison of Deontic Matrices for Small and Large B2C Exemplars

There are some interesting differences between Figures 5 and 6 (some rather intuitive, others much less so). Most notably:

- Small projects see no need to undertake a feasibility study, whereas this is optional for large projects
- Getting business approval is seen as optional for small projects, but highly recommended for large projects
- In small projects, there seems to be no need to identify the *sources* of the requirements, although, of course, *Task: Identify user requirements* is mandatory.
- Small projects use no tasks focussed on distributed computing
- Architectural design is optional for small projects, mandatory for large
- All the tasks associated with the *Component Selection Activity* as seen as of low priority (D for discouraged) whereas for larger projects they are recommended or, at worst, optional
- Creation of reusable components (for the future) is discouraged in small projects but encouraged in larger ones

- Optimization of the design is seen as mandatory for large projects, optional for small
- Choosing toolsets, hardware, project teams etc. is all but ignored in small projects (D for discouraged) whereas in large projects these tasks are recommended.
- Interestingly, the OPEN Task: *Specify individual goals* is forbidden in small projects but optional for large
- Small projects tend to be standalone whereas large projects tend to be integrated with other, pre-existing systems. This leads to values for OPEN Task: *Establish data take-on strategy* of F and M respectively – the opposite ends of the possibility spectrum.
- Subtasks of Task: *Model and re-engineer business process(es)* are less important for smaller projects
- Perhaps disappointingly (from a theoretical software engineering perspective), proponents of both large and small projects see little value in the OPEN Task: *Write manuals and prepare other documentation* (O and D respectively for large and small projects)

5. SUMMARY AND CONCLUSIONS

In this paper we have presented results, derived from several case studies, in the context of process engineering and process construction, results that are formulated as a set of deontic matrices to show the connection between activities and tasks for two categories of software development projects, namely: small/medium business-to-customer (B2C) and large business-to-customer e-commerce projects. A comparison of the data for these two domains highlights some interesting differences.

These differences can be important to understand, particularly for project managers involved in these types of projects. This allows the processes that are being adopted to be customised most appropriately to the characteristics of the project.

Ongoing work will focus on broadening the data into related domains, allowing improved guidelines to be constructed regarding the selection of relevant tasks.

6. ACKNOWLEDGEMENTS

This is Contribution number 02/05 of the Centre for Object Technology Applications and Research (COTAR). We wish to thank David Varnes from Comtech and Tim Kanneigter from Standards Australia for providing the data used to compile these matrix exemplars.

7. NOTES

- 1 Deontic logic is the logic of duty or obligation. It describes logical relationships among propositions that assert that certain things may be, for example, morally obligatory or morally permissible.
- 2 We use the term “forbidden” to mean that it is inappropriate for a specific linkage. We could alternatively use as a synonym a word such as “unrelated” or “irrelevant”
- 3 Or if they purport to, the user should have his/her suspicions aroused.

8. REFERENCES

- Brinkkemper, S., Saeki, M. and Harmsen, F., 1998, Assembly techniques for method engineering, Procs. CAiSE'98, LNCS 1413, *Advanced Information Systems Engineering* (ed. B. Pernici and C. Thanos), Springer-Verlag, Berlin, 381-400
- Chroust, G., 2000, Software process models: structure and challenges, in *Software: Theory and Practice – Proceedings, IFIP Congress 2000* (eds. Y. Feng, D. Notkin and M.C. Gaudel), 279-286
- Firesmith, D.G. and Henderson-Sellers, B., 2002, *The OPEN Process Framework. An Introduction*, Addison-Wesley, 330pp
- Graham, I., 1995a, *Migrating to Object Technology*, Addison-Wesley, Wokingham, England, 552pp
- Graham, I., 1995b, A non-procedural process model for object-oriented software development, *Report on Object Analysis and Design*, 1(5), 10-11
- Graham, I., and Henderson-Sellers, B., and Younessi, H., 1997, *The OPEN Process Specification*, Addison-Wesley, 314pp

- Haire, B., 2000, Web OPEN: an extension to the OPEN framework, Capstone project, Faculty of Engineering, University of Technology, Sydney, 122pp
- Haire, B., Henderson-Sellers, B. and Lowe, D., 2001, Supporting web development in the OPEN process: additional tasks, Procs. 25th Annual International Computer Software and Applications Conference. COMPSAC 2001, IEEE Computer Society Press, Los Alamitos, CA, USA, 383-389
- Henderson-Sellers, B., 2002a, Agile or rigorous OO methodologies – getting the best of both worlds, Cutter IT Journal, 15(1), 25-33
- Henderson-Sellers, B., 2002b, Process metamodelling and process construction, Annals of Software Engineering (in press)
- Henderson-Sellers, B. and Edwards, J.M., 1994, *BOOKTWO of Object-Oriented Knowledge. The Working Object*, Prentice Hall, Sydney, 594pp
- Henderson-Sellers, B. and Unhelkar, B., 2000, *OPEN Modeling with UML*, Addison-Wesley, Harlow, UK, 245pp
- Henderson-Sellers, B., Simons, A.J.H. and Younessi, H., 1998, *The OPEN Toolbox of Techniques* Addison-Wesley, UK, 426pp + CD
- Henderson-Sellers, B., Bohling, J. and Rout, T., 2002, Creating the OOSPICE model architecture – a case of reuse, Procs. SPICE 2002, Venice, Palazzo Papafava, March 13-15 2002 (ed. T. Rout), Qualital, Italy, 171-181
- Lowe, D. and Henderson-Sellers, B., 2001a, Web Development: Addressing Process Differences, Cutter IT Journal, 14(7), 11-17
- Lowe, D. and Henderson-Sellers, B., 2001b, Impacts on the development process of differences between web systems and conventional software systems. Procs. *SSGRR 2001: International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, (L'Aquila, Italy, 6-12 Aug), Scuola Superiore Guglielmo Reiss Romoli, pp21.
- OMG, 2001, *OMG Unified Modeling Language Specification, Version 1.4*, September 2001, OMG document formal/01-09-68 through 80 (13 documents) [Online]. Available <http://www.omg.org>
- Ralyté, J. and Rolland, C., 2001, An assembly process model for method engineering, Procs. CaiSE 2001, LNCS 2068, Advanced Information Systems Engineering (ed. K.R. Dittrich, A. Geppert and M.C. Norrie), Springer-Verlag, Berlin, 267-283

Rupprecht, C., Fünffinger, M., Knublauch, H. and Rose, T., 2000, Capture and dissemination of experience about the construction of engineering processes, Procs. CaiSE 2000, LNCS 1789, *Advanced Information Systems Engineering* (ed. B. Wangler and L. Bergman), Springer-Verlag, Berlin, 298-308