Method Engineering and COTS Evaluation

B. Henderson-Sellers
Faculty of Information Technology
University of Technology, Sydney
PO Box 123, Broadway, NSW,
Australia
+61 2 9514 1687
brian@it.uts.edu.au

C. Gonzalez-Perez
Faculty of Information Technology
University of Technology, Sydney
PO Box 123, Broadway, NSW,
Australia
+61 2 9514 4477
cesargon@it.uts.edu.au

M.K. Serour
Faculty of Information Technology
University of Technology, Sydney
PO Box 123, Broadway, NSW,
Australia
+61 2 9514 4479
mserour@it.uts.edu.au

D.G. Firesmith
SEI
Carnegie Mellon University
Pittsburgh
USA
dgf@sei.cmu.edu

ABSTRACT
This position paper argues that a successful COTS evaluation process should be based on the principles of method engineering (ME). Following a brief description of an ME approach underpinned by a metamodel, some method fragments related to component-based software engineering are offered as the starting point for the creation of a complete suite of method fragments for future COTS evaluation processes.

Categories and Subject Descriptors
D.2.2 [**Design Tools and Techniques**]: Object-oriented design methods

General Terms
Management, Design, Standardization, Theory

Keywords
Method engineering, COTS, process, method, methodology.

1. INTRODUCTION
The evaluation of COTS is but one component of a software development method. COTS evaluation may involve several steps, several focussed tasks and supporting techniques with the results of the evaluation being documented in some sort of output derived by the evaluation process. Consequently, the challenge is to identify the appropriate methodological fragments to support COTS evaluation. Here, we propose an approach, method engineering, that offers significant infrastructure support.

In Section 2, we outline the method engineering approach and then, in Section 3, discuss how method fragments are created and stored in a repository. Some pre-existing component-based software engineering (CBSE) method fragments are introduced in Section 4 in order to initiate discussion on how they might form an initial set of descriptors for future COTS-focussed fragments. Section 5 concludes with proposals for some possible future directions.

MPEC'05 at ICSE'05, May 21, 2005, St Louis, Missouri, USA.

2. METHOD ENGINEERING
Method engineering is an approach in which a method (a.k.a. methodology) is conceived of not as a single intertwined and interdependent entity but as a set of disparate fragments [1-3, 24; 29]. Those fragments are usually first identified by dissecting existing "one-size-fits-all" methodologies and also frequently created "bottom-up" from software engineering theory [20]. The fragments, which ideally comply with an underpinning metamodel [12], are stored in a repository. This is effectively standardized, either by an independent body or by the repository "inventors" and made available to software development organizations (generally commercially-focussed).

The software development team, perhaps headed by a method engineer and/or a project manager, then has the challenge of creating a "personalized" method. They select the most appropriate method fragments from the repository and assemble them into a full-blown method that has been effectively created precisely to fit the needs of the organization, product and/or project.

Ways by which method fragments are, firstly, selected and, secondly, used in method construction have been evaluated by several authors e.g. [2, 7]. Some approaches found useful are the use of pre- and post-conditions and the use of deontic matrices [8, 15]. Other potential sources of ideas for construction rules include [2, 6, 17, 19, 21-24]. However, there still remains work to be

undertaken to ensure that such approaches gain commercial acceptance.

3. METHOD FRAGMENT REPOSITORY

3.1 Granularity Issues

A repository for method fragments is generally constructed by a group of methodologists and method engineers in which the individual fragments conform to some specific "in-house" rules. In the approach of Ralyté [20], each fragment (called by her a "method chunk") involves a tight coupling between a task and a technique. In contrast, the granularity assumed in the OPEN Process Framework (OPF) approach [7] is that a fragment may be, *inter alia*, either a task or a technique. This latter approach allows a many-to-many relationship to exist between task and technique method fragments whereas in the former, for example, a technique used by two tasks will be redundantly incorporated into two different method chunks. Maintenance and integrity of the chunks in the repository thus becomes a significant issue since autonomy is obviated.

In addition, in the OPF approach, which we follow here, each of the method fragments is created by direct instantiation from an entity in the predefined and standardized metamodel. Currently, the OPF metamodel is that devised by the OPEN Consortium e.g. [7] but current and future work will align it directly with a new Australian Standard [28] in this area.

3.2 The Three Layers of the OPF

Elements of the metamodel (Figure 1) of the OPF cover technical, process, product, organizational and human-oriented aspects. Instances of these elements and their subtypes are then created (Figure 2) to cover a wide range of software engineering applications.

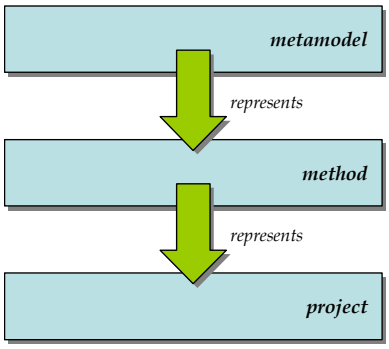

Figure 1 Representation of the metamodel, the method and the project levels.

The major elements in the metamodel are:

Work Unit. Work unit fragments describe what kinds of things are done (tasks and activities) and how these are accomplished (techniques)

Work Product. Work product fragments describe kinds of things input or output from work units.

Producer. Producer fragments document those people and tools being used for the creation and maintenance of work products.

Stage. Stage fragments (e.g. lifecycle, phase, milestone) are used to describe temporal aspects of an endeavour.

Language. Language fragments denote resources used to program, document designs or just plainly describe work products.

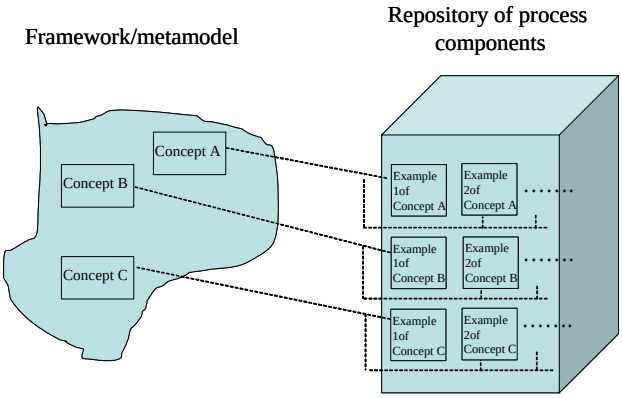

Figure 2 Instantiating metamodel elements in order to populate the repository with method fragments

Fragments generated as instances of these metamodel classes are stored in a repository located at the method level (Figure 1). Actual methodologies are then constructed from a selection of these repository-held fragments. Such a method may be configured specifically for use in a number of kinds of software or systems development, such as web-based applications [9, 15], organizational transition [14, 26], agent-oriented applications [13] and component-based software engineering (CBSE) [11, 27]. CBSE-focussed fragments are of relevance to COTS and are summarized below as a precursor to their extension and re-evaluation for use in COTS evaluations.

4. NEED FOR COTS FRAGMENTS

The fragments currently in the OPF repository do not currently fully support COTS. Indeed, we propose here a merger of the existing non-COTS, metamodel-based ME framework of the OPF with COTS evaluation ideas to be discussed in the 2005 ICSE/MPEC workshop.

As a first step, we would propose commencing with the existing OPF fragments devised to support the evaluation of components [11]. In summary, these CBSE-focussed method components are:

ACTIVITY: "Component selection" expands upon existing OPF fragments by the incorporation of CBSE-focussed ideas [18].

TASK: "Capture Business Requirement". To identify and analyze the business requirements for evaluating and acquiring COTS. This task should produce a checklist (see Technique: Checklist).

TASK: "Screen the candidate list of components". Vendors and available components are identified and screened against the business and development team's list of requirements.

TASK: "Evaluate the potential components". Full testing is undertaken of the potential candidates against pre-specified criteria.

TASK: “Choose appropriate components”. Based on the evaluation, a choice can be made based on a risk analysis and a trade-off analysis of costs and benefits.

TASK: “Integrate components”. The main focus on building systems from components and COTS software involves integration. Some useful support is also found in [25] and in Catalysis [5]

TECHNIQUE: “QESTA”. This is a technique for component evaluation devised by Hansen [10] that may be useful also for COTS evaluation. QESTA stands for Quantification, Examination, Specification, Transformation and Aggregation.

TECHNIQUE: “Checklist”. This is a useful technique to support the component selection tasks.

TECHNIQUE: “Compliance matrix template”. For tasks (see above) based on compliance evaluation, this template provides a useful starting point.

Using these CBSE-focussed fragments, we can analyze their utility, either directly or by extension, to COTS evaluation. For example, since COTS can readily be considered as a single (albeit large) component, the OPF Activity: “Component selection” would appear to be highly useful and relevant. Other tasks and techniques from those listed above should be similarly evaluated. In other COTS evaluation areas, no doubt there will be no pre-existing repository fragments – we anticipate that this workshop will help to identify these necessary additions to the OPF repository.

In addition to the method fragments listed above, the OPF can incorporate some method fragments defined in the context of the OOSPICE project [16, 27], which focussed solely on component-based software engineering. OOSPICE included the development of a method for object-oriented and component-based software development, and a number of method fragments were created. The OOSPICE metamodel is slightly different to the OPF metamodel, but straightforward mappings between the two can be easily done. The OOSPICE Task, Technique and Work Product metamodel elements have identical semantics to those in OPF with the same names. The semantics of the OOSPICE Process metamodel element are very close to those of the OPF Activity. Following these mappings, the following additional method fragments can be taken from OOSPICE and into OPF:

ACTIVITY: “Component Requirements Engineering”. The purpose of this activity is to elicit, analyse, specify and maintain evolving customer needs and requirements for the component.

ACTIVITY: “Component Architecture”. The purpose of this activity is to determine the logical and physical structure of the component in terms of its sub-components and mechanisms.

ACTIVITY: “Component Preparation”. The purpose of this activity is to prepare a component for integration in a particular application.

WORK PRODUCT: “Component Evaluation Document”. An evaluation conducted to ascertain the component's functional and quality characteristics so that decisions may be made about its suitability for integration into the product. The evaluation may vary from informal and cursory to formal and extensive. The evaluation would be expected to cover: functionality; quality characteristics such as reliability, robustness, performance; compatibility with the intended product; cost, including deployment costs; licensing restrictions.

WORK PRODUCT: “Component Acquisition List”. A list of components to be acquired. The list should describe each component and its requirements to guide attempts to acquire it.

In addition, a number of different tasks can be imported into the OPF from OOSPICE, namely “Analyse technologies”, “Develop vision statement for the component”, “Obtain requirements for the component”, “Analyse component requirements” and “Specify component requirements” (related to the “Component Requirements Engineering” activity); “Identify architecture styles and patterns”, “Determine logical elements of the component”, “Determine component infrastructures”, “Determine components”, “Make build/buy decisions”, “Establish traceability between component requirements and specification” and “Verify component architecture” (related to the “Component Architecture” activity); and “Evaluate component”, “Identify integration mechanisms for the component”, “Identify modifications”, “Implement modifications” and “Verify modified component” (all related to the “Component Preparation” activity).

5. FUTURE WORK

We have introduced here the notion that a method engineering approach, underpinned by a metamodel, can offer a firm and well-established base for the creation of method fragments for the evaluation of COTS. Standardizing COTS-related method fragments in this way permits researchers to evaluate different approaches to COTS evaluation within the same framework thus removing any biases due to incompatibilities of data sources. We therefore offer the OPF metamodel and repository as a starting point for future work on COTS evaluation.

6. ACKNOWLEDGMENTS

This is contribution number 05/03 of the Centre for Object Technology Applications and Research.

7. REFERENCES

[1] Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, **38(4)**, 275-280

[2] Brinkkemper, S., Saeki, M. and Harmsen, F., 1998, Assembly techniques for method engineering. *Proceedings of CAISE 1998*, Springer Verlag, 381-400.

[3] Brinkkemper, S., Saeki, M. and Harmsen, F., 2001, A method engineering language for the description of systems development methods (extended abstract), *CAiSE 2001* (eds. K.R. Dittrich, A. Geppert and M.C. Norrie), LNCS 2068, Springer-Verlag, Berlin, 473-476

[4] Constantine, L.L. and Lockwood, L.A.D., 1999, *Software for Use*, Addison-Wesley/ACM Press, New York, N.Y., USA, 579pp

[5] D’Souza, D.F. and Wills, A.C., 1999, *Objects, Components, and Frameworks with UML. The Catalysis Approach*, Addison-Wesley, Reading, MA, USA, 785pp

[6] Firesmith, D.G., http://www.donald-firesmith.com

[7] Firesmith, D.G. and Henderson-Sellers, B., 2002, *The OPEN Process Framework. AN Introduction*, Addison-Wesley, Harlow, Herts, UK

[8] Graham, I., Henderson-Sellers, B. and Younessi, H., 1997, *The OPEN Process Specification*, Addison-Wesley, UK.

[9] 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.

[10] Hansen, W.J., 1999, A generic process and terminology for evaluating COTS software, **in** *TOOLS 30* (eds. D. Firesmith, R. Riehle, G. Pour and B. Meyer), IEEE Computer Society, Los Alamitos, CA, USA, 547-551

[11] Henderson-Sellers, B., 2001, An OPEN process for component-based development, Chapter 18 in G.T. Heineman and W. Councill (Eds.) *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley, Reading, MA, USA, 321-340

[12] Henderson-Sellers, B., 2003, Method engineering for OO system development, *Comm. ACM* **46(10):** 73-78.

[13] Henderson-Sellers, B., 2005, Creating a comprehensive agent-oriented methodology - using method engineering and the OPEN metamodel, Chapter 13 in *Agent-Oriented Methodologies* (eds. B. Henderson-Sellers and P. Giorgini), Idea Group, Hershey, PA, USA

[14] Henderson-Sellers, B. and Serour, M.K., 2000, Creating a process for transitioning to object technology, *Proceedings Seventh Asia-Pacific Software Engineering Conference. APSEC 2000*, IEEE Computer Society Press, Los Alamitos, CA, USA, 436-440

[15] Henderson-Sellers, B., Haire, B. and Lowe, D., 2002, Using OPEN's deontic matrices for e-business, *Engineering Information Systems in the Internet Context* (eds. C. Rolland, S. Brinkkemper and M. Saeki), Kluwer Academic Publishers,

[16] Henderson-Sellers, B., Bohling, J. and Rout, T., 2004, Creating the OOSPICE model architecture – a case of reuse, *Software Process Improvement and Practice*, **8(1),** 41-49

[17] Hruby, P., 2000, Designing customizable methodologies, *JOOP*, **13(8)**, 22-31

[18] Kuruganti, I., 1999, A component selection methodology with application to the internet telephony domain, **in** *TOOLS 30* (eds. D. Firesmith, R. Riehle, G. Pour and B. Meyer), IEEE Computer Society, Los Alamitos, CA, USA, 552-556

[19] Martin, J. and Odell, J.J., 1995, Method engineering, Chapter 1 in *Object-Oriented Methods: Pragmatic Considerations*, Prentice-Hall

[20] Ralyte, J., 2004, Towards situational methods for information systems development: engineering reusable method chunks, *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education* (eds. O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic and S. Wrycza), Vilnius Gediminas Technical University, Vilnius, Lithuania, 271-282

[21] Ralyté, J. and Rolland, C., 2001, An assembly process model for method engineering, in K.R. Dittrich, A. Geppert and M.C. Norrie (Eds.) *Advanced Information Systems Engineering*), LNCS2068, Springer, Berlin, 267-283

[22] Ralyté, J., Deneckère, R. and Rolland, C., 2003, Towards a generic method for situational method engineering, *Procs. CAiSE2003* (ed. M.M. J. Eder), Springer-Verlag

[23] Ralyté, J., Rolland, C. and Deneckère, R., 2004, Towards a meta-tool for change-centric method engineering: a typology of generic operators, *Procs. CAiSE 2004* (eds. A. Persson and J. Stirna), LNCS 3084, Springer-Verlag, 202-218

[24] Rolland, C. and N. Prakash, 1996. *A Proposal for Context-Specific Method Engineering*. In *Procs. IFIP WG8 International Conference on Method Engineering*. Atlanta, GA.

[25] Seacord, R.C. and Nwosu, K.C., 1999, Life cycle activity areas for component-based software engineering processes, , **in** *TOOLS 30* (eds. D. Firesmith, R. Riehle, G. Pour and B. Meyer), IEEE Computer Society, Los Alamitos, CA, USA, 537-541

[26] Serour, M., Henderson-Sellers, B., Hughes, J., Winder, D. and Chow, L., 2002, Organizational transition to object technology: theory and practice, *Object-Oriented Information Systems* (eds. Z. Bellahsène, D. Patel and C. Rolland), LNCS 2425, Springer-Verlag, Berlin, 229-241.

[27] Stallinger, F., Dorling, A., Rout, T., Henderson-Sellers, B. and Lefever, B., 2002, Software process improvement for component-based software engineering: an introduction to the OOSPICE project, *Procs. Euromicro 2002 Conference*, IEEE Computer Society Press, Los Alamitos, CA, USA, 318-323

[28] Standards Australia, 2004, Australian Standard 4651-2004: Standard metamodel for software development methodologies, ISBN 0 7337 6195 X, 23 August 2004, Standards Australia International Ltd., Sydney, 72pp

[29] ter Hofstede, A.H.M. and T.F. Verhoef, 1997. On the feasibility of situational method engineering. *Information Systems.* **22**(6/7): p. 401-422.