Situational Method Engineering: Fragments or Chunks?

B. Henderson-Sellers[1], C. Gonzalez-Perez[2] and J. Ralyté[3]

[1]University of Technology, Sydney, PO Box 123, Broadway, NSW 2007, Australia
[2]European Software Institute, Parque Tecnologico #204, 48170 Zamudio, Bizkaia, Spain
[3]CUI, University of Geneva, Rue de Général Dufour, 24, CH-1211 Genève 4, Switzerland

Abstract. In Situational Method Engineering, the concepts of both fragment and chunk have been proposed as the atomic element. These are examined here in terms of their conceptual integrity and in terms of how they may be used in method construction, drawing parallels to show that the similarities are more than the differences. The new ISO/IEC 24744 standard metamodel is used as a conceptual framework to perform these mappings.

1. Introduction

High quality software development methods (a.k.a. methodologies [1]) can best be created by means of *construction* – identifying small elements of a methodology, variously called fragments or chunks, and putting them together for a specific situation [2]. Situational Method Engineering (SME) [3-5] provides a solid, theoretically sound basis for creating useful methodologies as well as giving the development team "ownership" of their methodological approach.

In the context of SME, there is a need to define the most appropriate atomic element from which methodologies can be constructed. There have been several proposals in the literature, variously known as a method chunk, a method fragment or a method component (or sometimes process component). In each case, the overall definition relies on an appropriate metamodel. In this paper, we examine the various proposals for an appropriate atomic element: fragment or chunk? Throughout, we use the ISO/IEC [6] Software Engineering Metamodel for Development Methodologies International Standard 24744 as an underpinning theory that assists us in identifying similarities and differences in these two approaches to SME.

2. Method Fragments and Method Chunks

A *method fragment* is an atomic methodological element *generated* from a metamodel – usually by instantiation. Many authors (and metamodels) discriminate between two kinds of method fragments: process fragments and product fragments [5,7]. The fragments generated, for instance, in the OPEN Process Framework [8] are each instantiated from a single class in the metamodel and are weighted towards specifying process elements (e.g. a kind of task or technique), whereas fragments that could be

extracted by sub-setting from OMT, OOSE or UML are more likely to be product-focussed fragments [9] (a kind of diagram, document or other work product). A third kind of fragment, producer-focussed, is missing from some approaches but present in the OPF [8], SPEM [10] and ISO/IEC 24744 [6] metamodels. A producer-focussed metaclass provides support for modelling the people involved in software development, the roles they play and the tools they use and is critical for creating a quality situational method. We therefore recommend its inclusion in future versions of chunk models. Its inclusion in a configuration of a triad of product, process and performer would, however, cause serious maintenance and consistency problems and raises concerns about chunk modelling (see below).

The fact that method fragments are not encapsulated together into chunks does not mean that there are no relationships between them. All the fragment-based approaches utilise some kind of association between process- and product-oriented fragments to capture the appropriate dependencies. This is best illustrated by ISO/IEC 24744, which models this relationship as a complete class, named *ActionKind*, representing a single usage event that a given process fragment exerts upon a given product fragment, thus providing clearly specified connectivity.

In contrast to the process-only or product-only fragment, other authors prefer the concept of a method chunk [2,9] in order to emphasize the more constructive, collection notion. Here, a chunk is the combination of a process fragment (also called a guideline) plus a product fragment (but omitting producers). The chunk captures the guidelines allowing production of the work product in the process portion of the chunk together with definitions of the concepts used in the product part [9]. Once a chunk is selected according to the methodological needs, the evaluation process, based on similarity measures [2], retrieves the method chunk – process plus product – and incorporates it to the methodology being constructed.

Finally, since chunks can be at any granularity, it is argued [9] that a full methodology itself can also be regarded as a chunk. This is similar to the model adopted more recently in SPEM Version 1 [10] in which a *Process* is modelled as a special kind of *ProcessComponent*. However, while this could work for fragments, since, by definition, a chunk is *one* process-focussed fragment plus *one* product-focussed fragment, then a full software development methodology cannot be envisaged as being a combination of one process-focussed fragment plus one product-focussed fragment, except at the most abstract level i.e. not in the endeavour domain where a methodology is enacted on a specific (situational) project.

3. The Pros and Cons of Fragments and Chunks

There has been much debate about the efficacy of a method chunk as compared to a method fragment for SME. In essence, as noted above, a method chunk is a conceptual combination of two method fragments: one process-focussed fragment and one product-focussed fragment. The advantage of such a combination is argued to be the speed of usage insofar as there are often a smaller number of chunks required for any specific situation and hence a small number that need to be located from the repository. Offsetting this to some degree is the fact that many of these chunks may

contain the same process part. In other words, there is a potential disadvantage as a result of the fact that such a process-product linkage is neither one-to-one nor unique in real-life scenarios. Indeed, if all such linkages *were* one-to-one, then the flexibility of method construction offered by SME would be totally redundant since everything would be "hard-wired". For instance, some techniques and work products can be used with more than one task such that several method chunks may contain the same product part but a different process part [9]; some tasks have multiple output products (one to many); some tasks modify existing products or have multiple inputs – and there are other examples in industry situations where a one-to-one linkage is not viable. When such many-to-one situations occur, a separate one-to-one chunk for *each* specific configuration needs to be created such that for instance, there is one chunk for one process fragment plus one product fragment; a second chunk for the same process fragment but with two different output product fragments, a third one for three outputs and so on.

Conversely, when the conceptual model is based on method fragments, there is greater flexibility but an additional effort in locating (in the repository) the best method fragments to link together, although a generic concept such as *Conglomerate*, found, for example, in ISO/IEC 24744, easily allows complex structures, such as a method chunk (with a process and a product part) to be constructed.

A second difference in fragment- and chunk-based approaches is the expression of the relationships between the product and process fragments/parts. In the fragment-based approaches the relationships between process- and product-oriented fragments are clearly specified by defining the type of action the process fragment is exerting on the product fragment. These relationships are mainly used to find the right couple of fragments (product fragment and process fragment).

In the chunk-based approach the relationship between the process and product parts of a chunk doesn't have the same role as it is not necessary to search for product and process parts separately. However, it is expressed by the chunk's *Intention*. For example, the intention of a chunk: "Create a Use Case model" states that the process part provides guidelines "to create" the product "a use case model". The intention is one of the parameters used to select the appropriate method chunks in a given situation.

Despite these differences, fragment-based and chunk-based approaches share a number of commonalities. To start with, both acknowledge the need to capture information about the situation where usage of any particular method component may make sense. In fact, this is a crucial aspect of *situational* method engineering; hence its name. Chunk approaches implement this via the chunk interface plus descriptor, which centralise situational information in a single place. In ISO/IEC 24744, as an example of a fragment-based approach, information has been modularised using different criteria, and situational information is distributed across different classes. First of all, the *Guideline* class is designed to capture information about where and how a method fragment (or collection thereof) can be used. Secondly, the *MinCapabilityLevel* attribute of the *WorkUnitKind* class captures the minimum capability or maturity level at which a particular process-oriented fragment is meant to be used, thus contributing to the establishment of a methodological situation.

Another similarity between fragment-based and chunk-based approaches is related to capturing information that may complement the specification of a method

component, such as bibliographic references. The chunk approach manages this through chunk descriptors, while ISO/IEC 24744 implements it through classes such as *Reference* and *Source*.

4. Summary and Conclusions

In the context of Situational Method Engineering (SME), we have evaluated the definition and descriptions of the atomic elements that are stored in a method fragment repository by contrasting the models for a method fragment, which depicts either a solely process-focussed concept, a product-focussed concept, or indeed (although not discussed in detail here) a producer-focussed concept, with that for a method chunk, which is a combination of a single process-focussed fragment with a single product-focussed fragment. We have identified some possible constraints on how chunks are retrieved from the repository, although this can depend significantly on the tool being used. Despite such concerns, chunks and fragments have much in common. They both take great care to capture information about the situation in which they can be used – critical for their employment in SME.

Throughout this analysis, we have used the new ISO Software Engineering Metamodel for Development Methodologies [6] as a means of providing a theoretical underpinning for our identification of similarities between the chunk and fragment approaches and for the mappings between them.

References

1. Jayaratna, N., 1994. *Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework*: McGraw-Hill.
2. 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.
3. Kumar, K. and R.J. Welke, 1992. *Methodology Engineering: a Proposal for Situation-Specific Methodology Construction*, in *Challenges and Strategies for Research in Systems Development*, W.W. Cotterman and J.A. Senn (eds.). Wiley: Chichester, UK. p. 257-269
4. ter Hofstede, A.H.M. and T.F. Verhoef, 1997, On the feasibility of situational method engineering. *Information Systems.* **22**(6/7), 401-422.
5. Harmsen, A.F., 1997, *Situational Method Engineering*, Moret Ernst & Young
6. ISO/IEC, 2007, *Software Engineering. Metamodel for Development Methodologies. ISO/IEC 24744*: International Standards Organization / International Electrotechnical Commission.
7. Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, **38(4)**, 275-280
8. Firesmith, D.G. and Henderson-Sellers, B., 2002, *The OPEN Process Framework. An Introduction*, Addison-Wesley, 330pp 2002
9. 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
10. OMG, 2002, *Software Process Engineering Metamodel Specification*, formal/2002-11-14.