Journal of Universal Computer Science, vol. 16, no. 3 (2010), 424-478
submitted: 31/3/09, accepted: 28/10/09, appeared: 1/2/10 © *J.UCS*

Situational Method Engineering: State-of-the-Art Review

Brian Henderson-Sellers
(University of Technology, Sydney, Australia
brian@it.uts.edu.au)

Jolita Ralyté
(University of Geneva, Switzerland
Jolita.Ralyte@unige.ch)

Abstract: The situational method engineering (SME) literature is surveyed and a synoptic evaluation presented in the context of formalizing and regularizing the conceptual framework and underpinning theory. Metamodels proposed for use in SME are evaluated as well as high-level process models for method construction. Method fragments and method chunks are then described formally followed by their identification and creation (from existing methods, from scratch or from past usage). Method creation is then analyzed in terms of various processes for constructing a full methodology from the method fragments/chunks. In particular, we contrast the use of the "map" technique and of the "deontic matrix" technique. The survey is concluded with an evaluation of some ideas on method tailoring and the emerging research on quality evaluation applied to SME.

Keywords: Design, Situational method engineering, methodology, software engineering, information systems, metamodels, method fragments, method chunks, method components, method construction, method configuration, method tailoring
Categories: D.2.2

1 Introduction

Method engineering (ME) and situational method engineering (SME) focus on formalizing the use of methods for systems development. The broader term, method engineering, is defined as the engineering discipline to design, construct and adapt methods, techniques and tools for systems development, a definition analogous to the IEEE definition of software engineering [Brinkkemper 06]. A major component of ME is situational method engineering, which encompasses all aspects of creating a development method for a specific situation (and excludes topics such as comparing methods and method knowledge infrastructures).

Method engineering and situational method engineering focus not on the acquisition of a ready-made method from some supplier (vendor or book-writing methodologist) but on the in-house construction of an organization-specific or project-specific methodological approach. This construction is accomplished by first selecting pieces of method (variously called method fragments or method chunks) that have been already created and stored in a repository or methodbase. The source of these stored fragments and chunks is not critical to the use of SME by practising

software developers. They may be "carved out" of other pre-existing methods or conformant to a standardized metamodel.

In this paper, we survey the topic of situational method engineering, a task last undertaken by [Tolvanen 96, Tolvanen 98]. SME is a solution offered to the problem of the selection of the "most appropriate" methodology for an organization and/or its projects. Authors have repeatedly noted that contemporary methodologies are not well suited to practice e.g. [Lyytinen 87, Glass 03], while [Avison 96] notes a backlash against formal software development methodologies. Others see process adoption as a "waste of time" [Baddoo 03], although, from a practical viewpoint, [Cockburn 00] argues that it is both appropriate and necessary for an organization to have available to it a suite of methodologies. [Avison 03] suggest that such disillusionment with the application of methodologies, which typically have historically failed to address important social, political and organizational factors, might lead to a "post-methodology" era in which there are no controls, standards or training. Since the one-size-fits-all methodology is now generally regarded as unattainable e.g. [Brooks 87, Avison 91, Kumar 92, van Slooten 93, Vessey 94, van Slooten 96, ter Hostede 97, Glass 00, Fitzgerald 03, Wistrand 04, Glass 04], alternatives have to be sought, particularly ones that take into account the human and organizational elements [Constantine 94]. Situational method engineering, including method construction e.g. [Henderson-Sellers 04b], possibly supplemented by method tailoring/customization (sometimes called method configuration [Karlsson 04, Ågerfalk, 04, Bucher 07]), is the current most optimistic route and forms the topic of this survey. This approach offers the practising software development professional the ability to follow a methodology that is precisely attuned to their individual needs and for which they feel some kind of "ownership" e.g. [Henderson-Sellers 05b]. Industry uptake is, however, currently slow (e.g. [Rolland 09]) partly because the costs of SME are perceived (incorrectly) as being larger than those of using an off-the-shelf methodology marketed by a large vendor or consulting company. This paper thus addresses both academic and industry-relevant topics that we hope will accelerate both the theoretical understanding of SME and its industry uptake. Indeed, the two most significant challenges for the SME community are the rate of industry adoption and how to automate the method construction process.

In the remainder of this paper, we first define the terminology used in SME (Section 2) and in Section 3 give a brief overview of the topic. In Section 4, we introduce metamodels, process models and other formalisms and in Section 5, we describe what is meant by a method chunk and method fragment and see how these can be formalized. In Section 6, we review the approaches to chunk/fragment identification and construction followed, in Section 7, by approaches to method construction from these fragments. Method tailoring is the topic of Section 8 as being applicable to either preformed methods or recently constructed methods. Section 9 offers some brief conclusions and directions for future research.

2 Terminology

Before progressing with our SME review, we must note that, in the field of process and method engineering, the terminology is often differently used between authors. There are three "key" high-level terms: method, methodology and process. A

(software/systems development) method can be defined as an approach to perform a software/systems development project, based on a specific way of thinking, consisting, *inter alia*, of guidelines, rules and heuristics, structured systematically in terms of development activities, with corresponding development work products and developer roles (played by humans or automated tools) (adapted from [Brinkkemper 06] – see also [Henderson-Sellers 95]). While the etymology of "methodology" gives its definition as the study of methods, it is in widespread and common usage meaning "method" [Jayaratna 94]. Furthermore, [Berki 04] gives credence to this second meaning (a meaning also given in many modern dictionaries e.g. the American Merriam-Webster Dictionary: [Cockburn 00]). For the purposes of this survey, we will indeed take the words method and methodology as synonyms, since we will not need the term methodology in its etymological sense.

The difference in meaning between the words "process" and "method/methodology" is harder to pin down. In general terms, a process is a way of acting, of doing something. Thus the way you relocate yourself from home to the work environment follows some (usually predefined – or least practised and often repeated) process. Thus, process is intangible and may be used in different situations at different granularities. However, to complement a process, there are other things that a software developer must be cognizant of: in particular, the work products produced and consumed and the people and tools involved in that production and consumption. Time sequencing is also of significant interest and concern. The word we will use here for this overall combination will be methodology or method[1]. In other words, a method(ology) encompasses absolutely everything needed for software development – [Cockburn 00] calls this a "Big-M methodology", [Glass 00] the "capital-M Methodology". Many authors use the description of an overall methodology as having two (often intertwined – or at least interdependent) aspects: product and process e.g. [Rolland 99], to which should be added an oft-neglected third: the people focus.

Another viewpoint is that the process describes what is actually done in real time with a real team on a real project. This is particularly the case in the capability assessment field where the focus of a capability assessment is *the process as it is performed* e.g. [Paulk 93, Dorling 93, ISO/IEC 98] – although often, for example in ISO12207 [ISO/IEC 95], processes are described as being at a smaller granularity and defined solely in terms of purpose and outcomes. Each process focusses on what is input to that process and what is output i.e. it is seen as a "transformation engine". It should be noted that there is both a static and dynamic aspect to such a notion of process i.e. the process (static enactment) whereby real developer's names, deadlines, deliverables replace the generic placeholders in the process model and the dynamics of the process as it is actually enacted [Greenwood 01] call the static enactment a "process model instance" and the dynamic enactment "process performance". Others talk of process models e.g. [Finkelstein 94, Gnatz 01, http://www.opfro.org] and process reference models [ISO/IEC 95, ISO/IEC 98], in contrast to the process as enacted on an individual endeavour, e.g. project. It is the description of the process model or process reference model that is typically documented in books, in reports or

[1] Strictly speaking we should say "software development methodology" since clearly the word "methodology" can be applied in a wide range of human endeavours.

on a website (although it is often labelled "process" rather than "process model"). A methodology is then a collection of these processes (perhaps as few as one) together with the product and people elements necessary, although sometimes it is just the process collection (process model) that is labelled as a method or "methodology" e.g. [Berki 04].

Together, this gives a multiple-layered model in which process, process model (and process reference models), method(ology) and metamodel can be depicted as suggested in Figure 1.

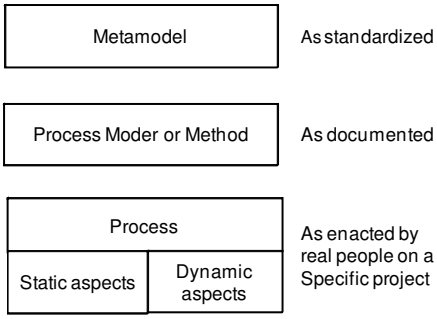

Figure 1: The "layers" of process and method terminology (after [Henderson-Sellers 06a]) Copyright LNI. Reproduced by permission.

In SME, the process/method is made up of smaller pieces (e.g. fragments, chunks). As noted by [Ralyté 01a, Ralyté 01b], the term *method fragment* was coined by [Harmsen 94] (and also by [Brinkkemper 96]) by analogy with the notion of a software component – see also [Saeki 93, Rolland 96]. It can be regarded as an atomic element of a method. In contrast, a *method chunk* is a combination of one process-focussed fragment plus one product-focussed fragment (see discussion in [Henderson-Sellers 08]).

It is well established that fragments or chunks, once derived, should be stored in a methodbase [Saeki 93, Brinkkemper 96, Harmsen 97, Rolland 98b, Ralyté 99, Ralyté 01a, Ralyté 01b], the structural aspects of which are discussed by e.g. [Harmsen 94]. When these are copied out of the database for a specific method construction exercise, project characteristics must be taken into account as contingencies that help determine exactly which fragments are (or are not) appropriate to present requirements. Figure 2 depicts simplistically how these elements are inter-related. Further discussion of the details of each of these three elements is found in succeeding sections.

[Rolland 96] stress the need to include knowledge about the *context of use* of the method fragments in a formal way within the methodbase. Context is here defined as the pair <*situation, decision*>. This means that the knowledge stored along with each chunk describes the situation in which it is relevant and the associated decision that can be made in such a situation. [Bucher 07] alternatively define "situation" as being a combination of context and project type. Both these formalisms represent the contingencies that must be taken into account in method construction.

Finally, and especially relevant for tool support, such as CAME or Computer Assisted Method Engineering tools e.g. [Tolvanen 98, Saeki 03] or metaCASE tools such as MetaEdit e.g. [Rossi 95, Rossi 98, Kelly 96] is the idea that, in turn, the

process model (or method) can itself be modelled. This is generally called the *metamodel*. The use of metamodelling in method engineering is discussed in detail in, e.g. [Henderson-Sellers 02b, Henderson-Sellers 06a, Greiffenberg 03, Gonzalez-Perez 08b] and, here, in Section 4.1. Note that in this paper, names with lead capitals refer to classes in the metamodel (often names ending in "Kind").

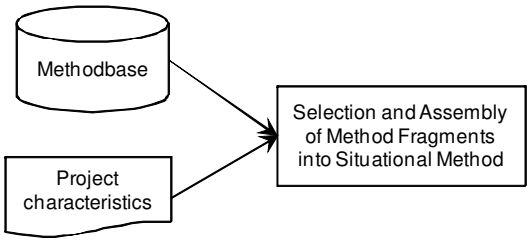

Figure 2: Engineering a situational method from the elements in the methodbase taking into account the prevailing situation including project characteristics, overall context and other contingencies

Another characterization that of levels of granularity, is discussed by several authors. [Ter Hofstede 97] identify three levels, which they call (somewhat confusingly in the light of the definitions above) product level, model level and component level. In [Rolland 96], granularity levels are said to be context, trees and forests of trees. These permit coarse granular chunks such as the whole of OMT [Rumbaugh 91] to be regarded as a single component. Some are reusable as is and some need instantiating first – see later formalization in [Gonzalez-Perez 05b].

3 Method Engineering Overview

Prior to the adoption of object technology (the underpinning rationale for all of today's method engineering literature and practice), the TAME project e.g. [Basili 87, Jeffery 88, Oivo 92] aimed to deliver a tool to support some amount of tailoring (i.e. minor modifications to support local conditions) of a pre-existing process or methodology. The last two decades have seen the emergence and establishment of object technology with its central tenets of encapsulation, information hiding and modularity. The notion of class that then emerges has become the standard way to thinking about and depicting method fragments and their underpinning metamodels. Method engineering has emerged as a potentially important component of software engineering in parallel to the acceptance of this "object-oriented paradigm".

Method Engineering (ME) was introduced as a term by [Bergstra 85] and then again by [Kumar 92] who named it methodology engineering; but [van Slooten 93, Brinkkemper 96] strongly recommend changing this to method engineering, a term that has been generally accepted since [Brinkkemper 96]'s definition of method engineering is useful here: "Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems." Interestingly, [Glass 00] equates the ME approach to an *ad hoc* approach in that the correct meaning of *ad hoc* is "suited to purpose" or "tailored to

the problem at hand". (This is different to the use of *ad hoc* in some of the strategies described below.) Situational Method Engineering (SME) is ME applied to a particular situation, often specific project characteristics (Figure 2). The (situational) method is thus created in-house, thus giving the software development team ownership of the resulting method e.g. [Henderson-Sellers 03]. Practical advice on such creation is offered in terms of process construction using a pattern-based approach. Explicit discussion of patterns is found in, for instance, [D'Souza 98, Ambler 98, Ambler 99, Hruby 00, Fiorini 01, Gnatz 01, Tran 05, Tasharofi 07] and implicitly in [Graham 97].

Since method knowledge is now fragmentized and highly structured, additional fragments tailored specifically to a single organization, or added as technology changes – for example, the recent emergence of web services e.g. [van de Weerd 05], agility e.g. [Cossentino 05, Qumer 07], agents e.g. [Henderson-Sellers 05a], product lines e.g. [McGregor 08], governance, risk and compliance [Gericke 09] and even the games industry [van de Weerd 07] – can be easily added to the fragment repository (a.k.a. methodbase). The challenges of method engineering relate to the various elements of this overall process viz. how to create method fragments, how to store and retrieve them, how to assemble a full methodology from the fragments and how to formalize the repository-held fragments (typically by use of a metamodel). These are all major topics in this review.

From the practical viewpoint, the lack of empirical data is noted by [Tolvanen 96]. They urge the collection of data through longitudinal studies augmented by user satisfaction surveys, as well as the efficacy of action research (see also [Tolvanen 98]). Notwithstanding, an increasing number of case studies have been published to describe specific example applications and domains of applications. For example, [Rolland 02] addressed the Lyee methodology; [Aydin 02] addressed DSDM (Dynamic Systems Development Method: [Stapleton 97]) while [Zowghi 05] derive a requirements engineering (RE)-focussed process based on the OPEN Process Framework (OPF) [Firesmith 02, Henderson-Sellers 04b]. Another RE-focussed study [Jiang 08], whilst not primarily one in the SME domain, provides potentially useful ideas for fragment selection. The domain of web content management has been analyzed recently by [van de Weerd 05] while a series of papers e.g. [Serour 02, Serour 04a, Serour 04b] describe how SME was used in several industries in Sydney, particularly a legal publisher and an e-government IT support group. Similar studies, performed with industry in Slovenia, are reported in [Bajec 07a], with health care specialists in Australia by [Waller 06] and with three large Swedish IT companies by [Karlsson 07].

Many authors also discuss potential automation of the SME process e.g. [Saeki 03]. Following [Harmsen 94, Odell 95] suggests that any CAME (Computer-Aided Method Engineering) tool should support the following seven features:

- Definition and evaluation of contingency rules and factors,
- Storage of method fragments,
- Retrieval and composition of method fragments,
- Validation and verification of the generated methodology,
- Adaptation of the generated methodology,
- Integration with a meta-CASE tool,
- Interface with a methodbase.

A detailed evaluation of CAME tool creation, validation and utilization, while necessary in an industry usage context, is deemed outside the scope of this review. Useful references can be found in [Tolvanen 98] based on the MetaPHOR project [Lyytinen 94] and, for agent-oriented applications, in [Garcia-Ojeda 09].

Note that process modelling languages themselves e.g. [Chou 02, Niknafs 09] are also deemed outside the scope of this review. The process modelling literature tends to focus either on ways to use modelling to support flexibility in the enactment domain e.g. [Cunin 01] or on the description or visualization of processes e.g. [Stőrrle 01, Scott 01, Becker 03]. In contrast to such *descriptive* approaches, ME/SME attempts to formulate a *prescriptive* model [Rolland 95]. These correspond to the backward-looking and forward-looking models described in [Gonzalez-Perez 07].

Some of the challenges for situational method engineering include the creation of commercial strength methodbases, tools to support method construction, theory and tools to support quality evaluation of the constructed method, the availability of SME tools and their interface to other toolsets likely to be needed, knowledge of what works in what situation and why? The most recent document reflecting research issues in SME is the proceedings of the IFIP WG8.1 Working Conference, held in Geneva in September 2007 [Ralyté 07].

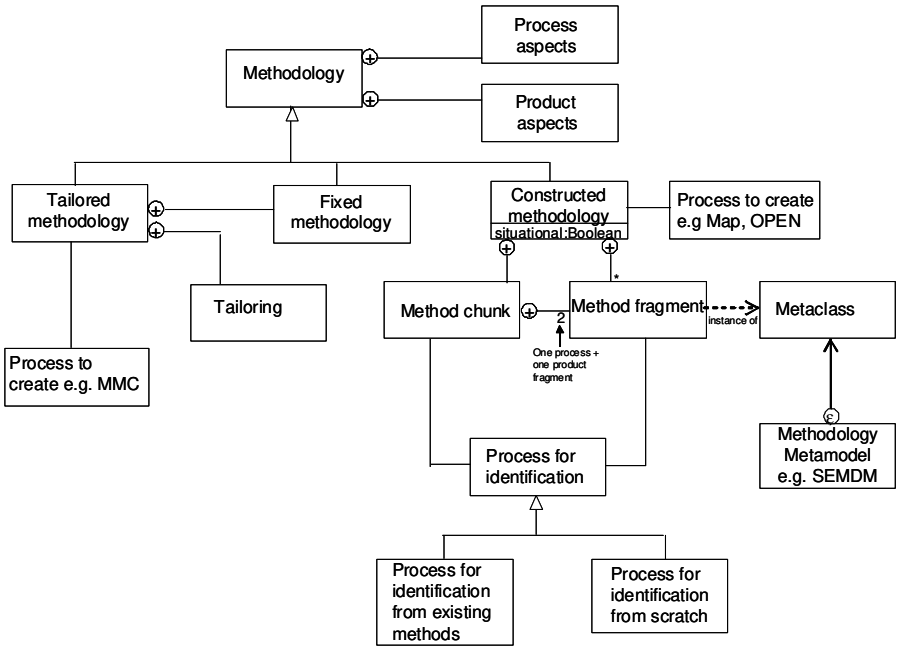

Figure 3: An overall high-level model of the SME approach, used to structure this presentation. It is not intended to be a complete metamodel. Note that the "plus in circle" and "epsilon in circle" symbols are the OML [Firesmith 97] icons for a configurational and non-configurational whole-part relationship respectively, used here since these relationships are not supported in UML [Barbier 03]

The overall approach to SME is synopsized in Figure 3 as an object-oriented class model, which also serves as an architecture for this survey paper. Methodologies have both process and product aspects. They may be constructed, tailored or fixed. Here, we focus on the first two. Constructed methodologies are the main focus of the paper including processes by which such construction occurs and the "parts" from which the methodologies are constructed (chunks and/or fragments), typically defined in terms of a metamodel. Also covered here are the processes by which chunk creation, identification and storage support the overall aims of SME. Tailored methodologies similarly require information on processes for the actual tailoring activity.

4 Formal Aspects of SME

In this section, we look at how more formal techniques have been incorporated into situational method engineering. In Section 4.1 we discuss various metamodelling approaches at a variety of scales, from full method to single fragment descriptions. In Section 4.2, in contrast to the static models of Section 4.1, we outline several formal descriptions of ways of constructing methods, often using the notion of an underpinning metamodel. Section 4.3 notes some approaches that use other formalities such as Petri nets and process algebras.

4.1 Underpinning method metamodels

There are multiple dimensions to modelling. In particular, models can be "stacked" in terms of their abstraction level. This is readily seen in the 4-layer metalevel hierarchy (Figure 4) of the OMG (see e.g. [OMG 01]) – although it should be noted that the inter-level relationship of "instance-of" has recently been subject to some criticism e.g. [Seidewitz 03, Atkinson 01, Gonzalez-Perez 06a, Gonzalez-Perez 07].

The use of metamodels in general is recommended in [Madhavji 91, Rolland 95, Rolland 96, Tolvanen 96, Jarke 98] and was always the core underpinning for the OPEN Process Framework e.g. [Henderson-Sellers 96b, Graham 97, Firesmith 02]. Indeed, [Ralyté 03] refer to it as the "core technique in SME". Metamodels provide a means of defining the rules (for a modelling language or for a method) at a higher level of abstraction.

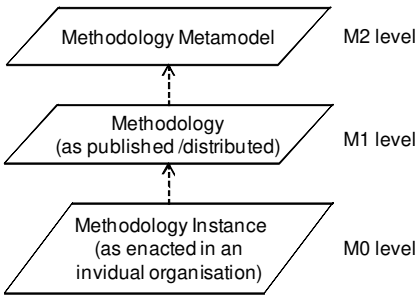

Figure 4: OMG's multi-level hierarchy – the bottom three layers adopted for process and method(ology)

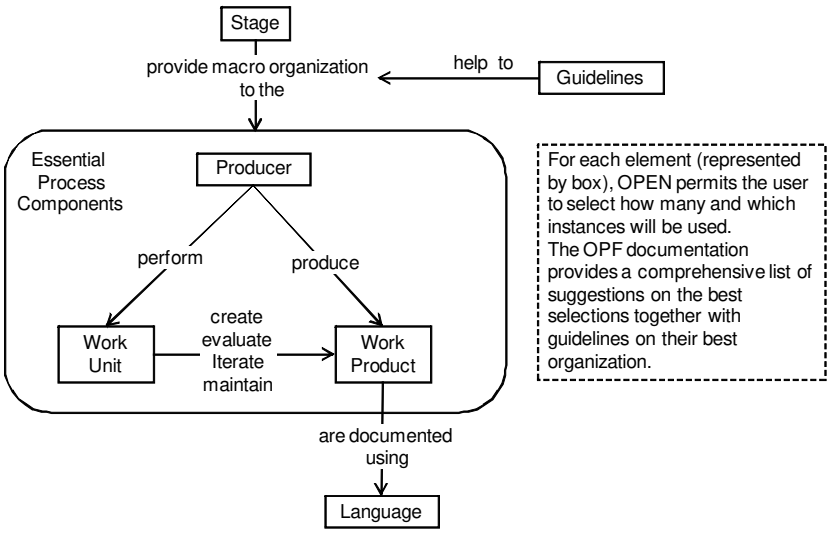

Figure 5: The 3+2 core classes of the OPF process-focussed metamodel (after [Firesmith 02]) Reproduced with permission from Pearson Education Ltd.

In the domain of object-oriented methodologies, metamodels were used first[2] to formalize a large number of OO methods of the early 1990s in the COMMA project [Henderson-Sellers 96a, Henderson-Sellers 98a]. This study was aimed at identifying commonalities prior to conciliatory discussions between methodologists (discussions that later eventuated within the Object Management Group (OMG) and led to the creation of the Unified Modeling Language). Metamodels then became, within the OMG, the underpinning rationale not only for the UML but also for the MOF (meta-object facility), SPEM (software process engineering metamodel) and MDA (model-driven architecture) standards initiatives. More recent standardization efforts in the use of metamodels to underpin methodologies (and hence ME/SME) have been seen in Australian Standard 4651 [Standards Australia 04] and the ISO/IEC International Standard 24744 [ISO/IEC 07] a.k.a. SEMDM (Software Engineering. Metamodel for Development Methodologies).

The *core* of many process-focussed metamodels (e.g. OPF, SPEM) comprises just three classes: Producer, Work Unit and Work Product (although the actual names may differ). These are illustrated in Figure 5, which depicts the OPF core metamodel, augmenting these core classes by two significant supporting classes: Stages (to describe calendar time) and Language (to describe ways of documenting the various

[2] Although an earlier study [Hong *et al.*, 1993] purports to use metamodelling, its so-called meta-process and meta-data models are more closely identifiable as process reference models i.e. belonging to the middle level of Figure 1 or the method level of Figure 5.

models) – a model used in [Niknafs 09] in their analysis of process modelling languages. In the OPF and SPEM, method fragments are generated by instantiation[3] from these top levels classes and their subclasses. Each fragment is either a producer-focussed fragment, a work-unit-focussed fragment or a work-product-focussed fragment. In AS4651 and ISO/IEC24744, the layered metamodel architecture is slightly different (Figure 6), incorporating powertypes [Odell 94, Gonzalez-Perez 06a, Gonzalez-Perez 06b, Gonzalez-Perez 08b]. Thus, the fragments are generated by powertype instantiation (rather than "regular" instantiation). Since a powertype combines instantiation semantics with generalization semantics, it allows some attributes of the powertype pattern present in the metamodel domain (in the example shown in Figure 7, Document plus DocumentKind comprise the powertype pattern) to be inherited by the class[4] in the method domain with other attributes having actual values allocated to them. In Figure 7, for example, the attributes Title and Version of Document are inherited unchanged and become attribute specifications in RequirementsSpecificationDocument. In contrast, the attributes Name and MustBeApproved of DocumentKind, specified in the metamodel domain are instantiated to become values (Req.Spec.Document and Yes respectively) of the RequirementsSpecificationDocument. These latter values apply to *all* requirements specification documents, whereas those inherited from Document still need to have values allocated since such values are individualistic to projects in the Endeavour domain. Thus instantiation semantics between the clabject in the method domain to the object in the endeavour domain allow, in this example, the allocation of the value of "MySystem"Req.Spec. to the Title attribute of RequirementsSpecification Document and the value "Version 1.5" to its Version attribute. In summary, the advantage of powertype instantiation is that it permits some attributes, defined in the metamodel, to be given values in the method domain and others in the enactment domain [Gonzalez-Perez 06b] – a problem not solvable using the strict metamodelling approach exemplified in the OMG architecture of Figure 4. In addition, this powertype-based approach provides semantic integrity between the process and product aspects of a methodology, not possible when linking SPEM or OPF to a modelling language like UML.

[3] Although we keep with the familiar phrase "generated by instantiation", this is not strictly accurate. The model fragments are not instantiated from the metamodel but rather they are *conformant to it*.

[4] Actually a clabject, an entity with both a classlike and an objectlike nature: [Atkinson 98]

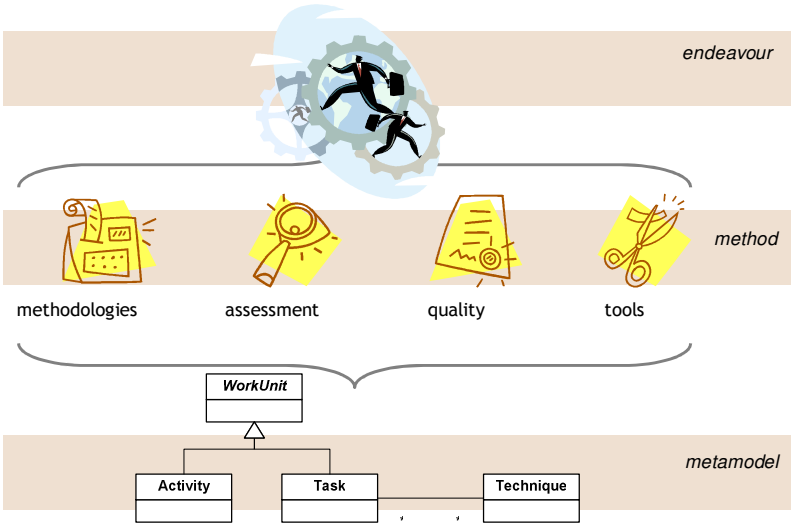

Figure 6: The three domains of AS4651 and ISO/IEC 24744 (This is inverted with respect to the OMG architecture shown in Figure 4 in order to stress the importance of people in the endeavour domain) and the fact that the metamodel provides the foundational level (after [Henderson-Sellers 06a]) Copyright LNI. Reproduced by permission

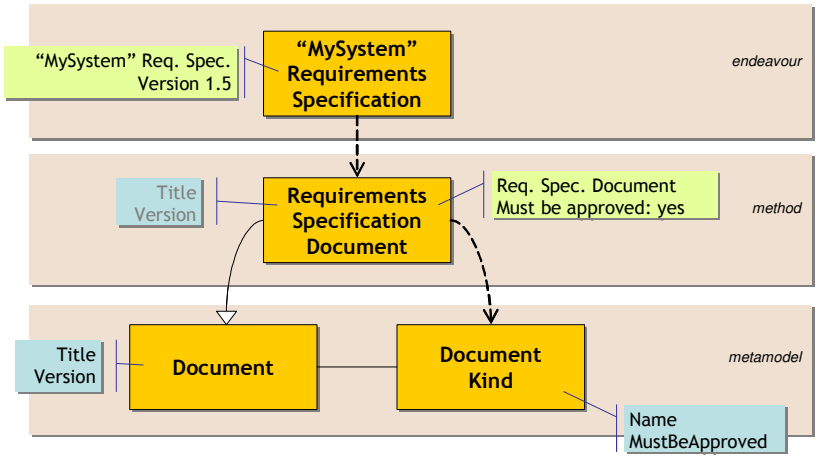

Figure 7: The use of powertype instantiation [e.g. in ISO/IEC 24744] to transmit attribute values to either the model domain or the enactment (endeavour) domain (after [Henderson-Sellers 06a]). Copyright LNI. Reproduced by permission

Although neither SPEM nor OPF have a class explicitly labelled "method fragment", such a class is essentially the abstract supertype of all the classes contained in the metamodel. For example, in AS4651 and ISO/IEC 24744, there is a class called Methodology Element which is a synonym for Method Fragment.

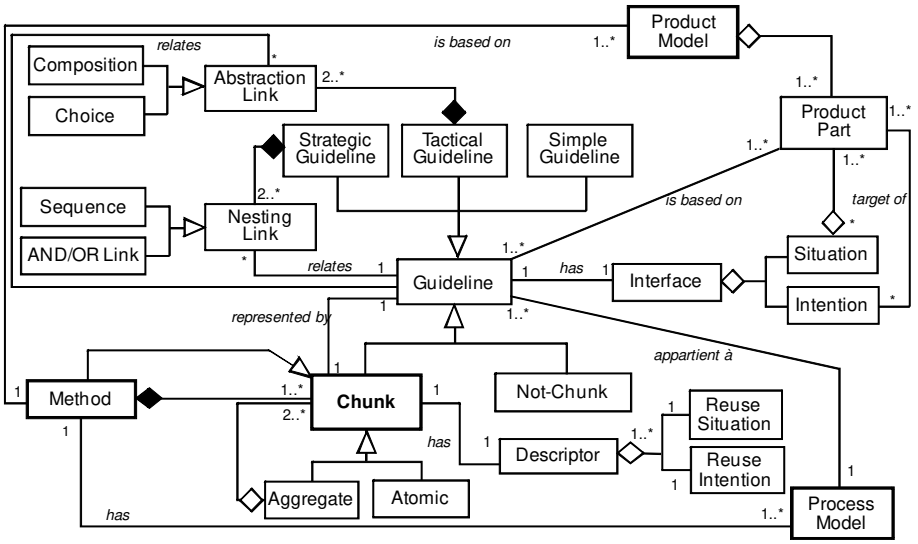

Figure 8: The method metamodel of [Ralyté 01b] Copyright Springer-Verlag 2001. Reproduced with kind permission from Springer Science and Business Media

An alternative approach is to define a method chunk (as opposed to a method fragment) as a combination of a process-focussed fragment and a product-focussed fragment. A number of metamodels for such an approach have been proposed – two examples are shown in Figures 8 and 9. In Figure 8 is depicted a generic ME/SME metamodel for a modular method – a method composed of a collection of method chunks [Ralyté 01b]. A Chunk has a ProductPart because it inherits its association to Guideline. The complement, ProcessPart, does not appear under this name. Instead, the ProcessPart appears under its synonym of Guideline. Guidelines may be categorized as either strategic, tactical or simple (see detailed discussion in Section 4.2 below). Interface refers to knowledge stored about the context of the use of the method chunk [Henderson-Sellers 07b]. Thus it consists of two parts, one relating to the situation (the pre-condition for usage) and one to the intention (the post-condition of the use of this chunk). The descriptor is only used to add information relevant to chunk retrieval, as elucidated in [Mirbel 06a, Mirbel 06b, Henderson-Sellers 07b].

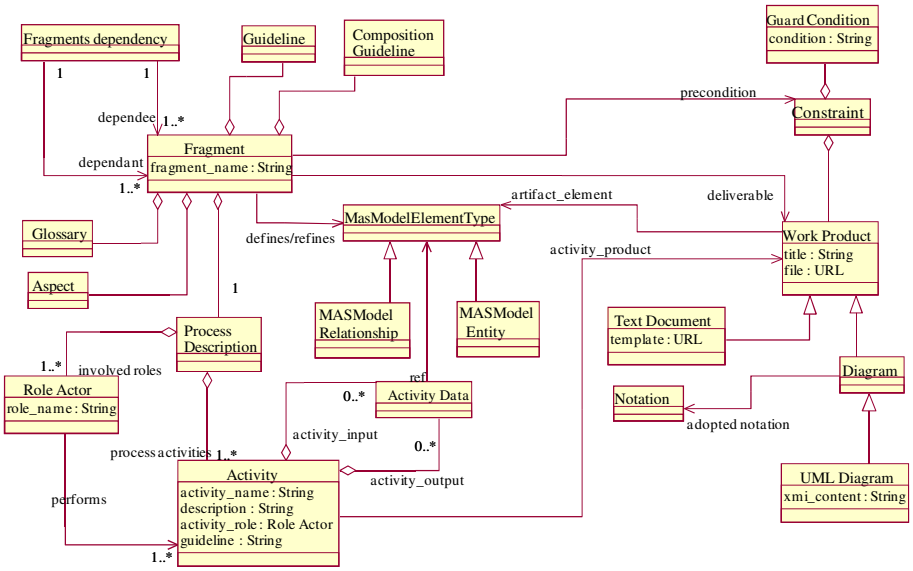

Figure 9: The metamodel of the FIPA so-called method fragment (actually a method chunk) (after [Cossentino 06])

While Figure 8 is independent of technology, Figure 9 is a chunk metamodel created specifically for agent-oriented software development under the auspices of FIPA (Federation for Intelligent Physical Agents – now a committee within the IEEE Standards responsibility). This metamodel describes a chunk (inappropriately labelled "Fragment" in Figure 9) as a combination of a ProcessDescription, a Glossary (a list of terms that facilitate the understanding of fragment concepts when applied to a context different from the one from which it was extracted), an Aspect (a textual description useful for detecting the field of fragment application, for instance a tool to be used to aid in the performance of an activity) and two kinds of guidelines: *Guideline* refers to the fragment as a portion of a process i.e. a set of rules providing a detailed description on how to perform an activity; and C*ompositionGuideline,* which describes the context from which it is extracted, indicating the reuse possibility for the fragment. The product aspect of a fragment is depicted by a directed association to WorkProduct.

It should be noted that both these chunk metamodels include a significant number of methodology elements outside of the scope of a definition of what a chunk itself is. Further discussion of method chunks and method fragments is found in Section 5 (and also [Henderson-Sellers 07b]).

4.2 Process models for method construction

Many of the processes involved in method engineering can be described by process models notated using the concept of a *map* [Rolland 99]. A map is described as a directed labelled graph consisting of nodes representing *intentions* and edges to represent *strategies*. An intention captures the notion of a task to be accomplished

whereas the strategy suggests the way in which this goal can be achieved. [Ralyté 01a] note that the core concept in a map is the *section*. This is defined as a triplet given by section = <*source intention, target intention, strategy*> or <I_i, I_j, S_{ij}>. A map is then a composition of a number of sections, expressed by the authors as map = ∑*section* = ∑<*source intention, target intention, strategy*> plus a *Start* and a *Stop intention* [Rolland 99].

Figure 10 depicts, stylistically, the elements of a map. Let us assume that there are three intentions (represented here by the nodes I1, I2 and I3) forming a fragment of a map in which intention I1 has already been achieved. The question is what intention is the next and by what strategy is it achieved. From the viewpoint of I1, the first problem is whether to select intention I2 or intention I3. Advice on this selection is needed. Let us say I2 is selected. We note that this can be achieved by one of two strategies, S12a and S12b – but which should be selected? Advice on this selection is needed. Once a strategy has been selected (say S12a), then advice is needed on how to enact the selected strategy.

These three pieces of required advice (said to embody *method knowledge*) are represented as *guidelines* [Rolland 99]. These are, respectively, an Intention Selection Guideline (ISG), associated with the source intention; a Strategy Selection Guideline (SSG), associated with a pair of intentions <Ii,Ij>; and an Intention Achievement Guideline (IAG), associated with a section (node pair plus strategy) (Figure 11). In the example of Figure 10, when the decision is being made regarding moving from node I1 to either I2 or I3, the advice is given as an ISG. Having made a selection (in the above example I2), the means to make the transition – a choice of several strategies – is given by an SSG. The actual enactment of the selected strategy (here S12a) also requires advice – from the IAG.

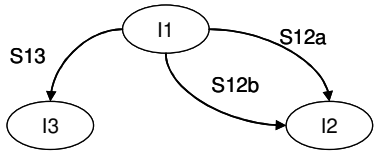

Figure 10: Stylized representation of a map fragment showing associated IAGs.

Both the ISG and SSG are navigational guidelines. It is the IAG that embodies potential subprocesses and can therefore itself be represented as an (embedded) map. Since an IAG shows how to realize the target intention from the source intention using the selected strategy [Ralyté 01b], it is said to depict "tactics". Indeed, [Ralyté 01a] suggest that it is the merging of the tactical aspects of guidelines with the strategic aspects of the map that provides most value to SME.

Thus, there is a recursive possibility of a section being refined as an entire map at a lower level of granularity [Ralyté 01a, Ralyté 04a] – somewhat akin to the levelling notion of data flow diagrams (Figure 12). Thus the strategy (S) linking two nodes is explicated by an IAG. This IAG, which represents a mini-process, can therefore itself be represented as a map. Thus at a lower level, the IAG itself is depicted as a set of nodes and edges (strategies). Then recursively each strategy in this lower level map

can be elaborated upon with an IAG which in turn could be represented as a map. And so on.

Guidelines embody method knowledge and are described in terms of a body that encapsulates this knowledge, together with a signature [Rolland 99]. The signature of a guideline (later renamed interface by this research group) is a combination of a situation and a relevant intention i.e. signature = <*situation, intention*>.

For the navigational guidelines (i.e. ISG and SSG), the relevant intentions represent the progress to or from the intention rather than the intention itself. The situational part refers to the product part(s) resulting from the achievement of the source intention i.e. it plays the role of a precondition. The body of a guideline can be executable, plan or choice with two different contextual relationships: composition or refinement. It describes how the chunk should be applied in order that the intention is achieved [Ralyté 04a].

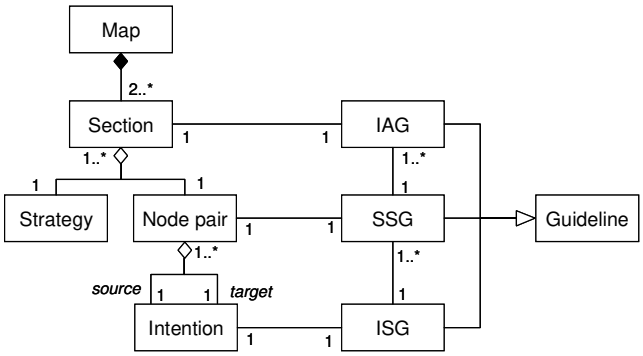

Figure 11: Metamodel depicting guidelines

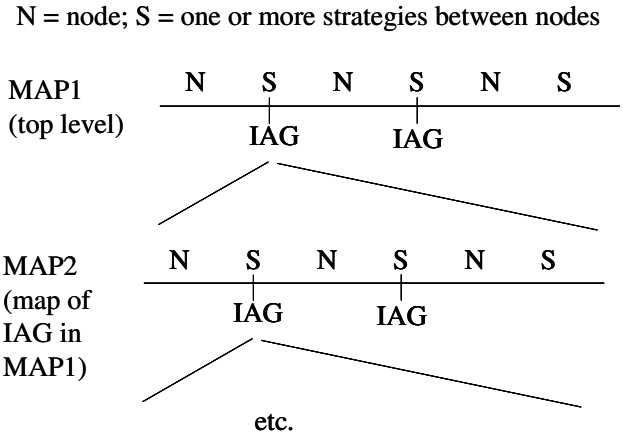

Figure 12: The notion of levelling within a map

In some apparent contrast is the partitioning of Guidelines into strategic, tactical and simple of [Ralyté 01b]. This suggests that two of the guidelines of Figure 11 (ISG

and SSG) are perhaps subtypes of Tactical Guideline (which uses the NATURE modelling formalism: [Jarke 99]) with IAG being a kind of Strategic Guideline, which itself is closely associated with the map concept, as suggested by [Ralyté 01b].

In summary, the map approach to visualizing and enacting a process model explicitly recognizes the role of *strategies* in realizing goals (a.k.a. intentions), provides a multi-thread and multi-flow, non-deterministic approach to process (and hence method) engineering in which dynamic SME is the rule rather than the exception [Rolland 99]. Indeed, it is entirely feasible to add new nodes and/or arcs (strategies) dynamically as the process map is enacted.

Whereas, in the "map" approach, the node represents a goal or an objective to be met, in the OPF's "deontic matrix" approach (described in detail in Section 7 in the context of process construction), the goal (or new state) to be achieved is represented as a Task (or an Activity at a coarser granularity). Achievement of Tasks is the result of the use of an appropriate Technique, which is linked to the Task using fuzzy, deontic values. These values play a similar role in the selection of tactics as does the IAG and its associated (sub-process) map(s) as described above. An initial appraisal of the mappings between these two approaches and UML activity-style diagrams is presented in [Seidita 07].

4.3 Other formalisms

Although it is generally acknowledged that all fragments should adhere to a metamodel definition, formal semantics are still needed, and not just for the structural and representational aspects. It is the *meaning* that must be correctly captured [ter Hofstede 97]. This could be done by using clear and concise natural language, a process algebra, temporal logic or Petri nets, for example [ter Hofstede 97]. To date there has been a greater focus on ways of modelling e.g. UML [OMG 01], GOPPR [Tolvanen 93] than on the people issues, as addressed, for example, by the "ways of working" of [Rolland 95].

The ER-based language MEL (Method Engineering Language) [Brinkkemper 96, Brinkkemper 01] anchors method descriptions in an ontology, especially useful for SME. It has basic operations to insert and remove fragments in and out of the methodbase, to retrieve fragments and to assemble them into a situational method. It also has syntactic constructs to compose a complex process from activities, such as sequencing, conditions, iterations, parallelism and non-determinism (for further details of MEL see Section 6.3).

5 Method Chunks and Method Fragments

In any discipline, agreed terminology (concepts, semantics, ontologies etc.) can increase the speed of acceptance in the wider practitioners' community. Software engineering in general and object technology in particular has been plagued by the lack of an agreed ontology e.g. [Firesmith 95, SEMAT 10]. In SME, one major area that needs rationalization is the correct use of the terms "method chunk" and "method fragment" [Ågerfalk 07].

As noted earlier (in Section 2), a fragment is an atomic part of a methodology represented by a class in the metamodel. Many authors discriminate between two

kinds of method fragments: process fragments and product fragments e.g. [Rolland 96, Brinkkemper 96, Punter 96, Harmsen 97, Brinkkemper 98, Rolland 99, Brinkkemper 01, Ralyté 01a, Ralyté 01b, Ralyté 04a, Ågerfalk 07]. These describe a single part of the method. On the other hand, others prefer the term *method chunk* [Rolland 96, Plihon 98, Rolland 98b, Ralyté 99, Ralyté 04a, Ralyté 01a, Ralyté 01b, Mirbel 06b] to emphasis the more positive[5], collection notion. *To them, a chunk is the combination of a process fragment (also called a guideline) plus a product fragment.* In [Ralyté 01b], a method chunk is thus a tightly coupled (process+product) representation[6]. This coupling of a single process part and a single product part uses the definitions of [Ralyté 99], which in turn expands the process-only focus of the definition of chunk in [Rolland 95, Rolland 98a]. More exactly, the product part of a method chunk includes all product elements necessary for the process part execution: the input and output product elements. Therefore, while the process part of a method chunk can be aligned with the notion of process fragment (work unit in the OPF definition), the definition of the product part is somewhat different from that of the product fragment. However, we can consider that the definition of the target product of a method chunk (the output product) corresponds to the notion of product fragment (work product in the OPF).

With these two agreed definitions (method fragment and method chunk) in mind, confusion can arise when the composite notion of a "chunk" is used but is referred to as a method "fragment" e.g. [Cossentino 06], then used by [Bucher 07], which in turn was adopted for use in a different software engineering domain by [Gericke 09]

The advantage of a combination of atomic "fragments" into a single "chunk" is argued to be the speed of usage insofar as there is often a smaller number of chunks required for any specific situation and hence a small number that need to be located from the methodbase. Offsetting this to some degree is the fact that many of these chunks may contain the same product part or more exactly same product elements. 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 (see later developments discussed below). 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; 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. With the assumption of a 1:1 connection between process and product part of a chunk, 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

[5] This use of positive language is echoed by Don Firesmith who recommends the term *method component* (see http://www.opfro.org/).

[6] The discussion here rests on the assumption that there is a 1:1 relationship between process part and product part, as often stated. This is in contrast to some metamodels, such as that shown in Figure 13, where the cardinalities are 1:m or even m:m (see also later discussion).

second chunk for the same process fragment but with two different output product fragments, a third one for three outputs and so on. In addition, even if this were legitimized[7] in some way within the chunk approach, a catalogue would be needed for the methodbase (i.e. the knowledge base that a method engineer would use in searching for appropriate chunks), which could rapidly fill up with partially duplicated, overlapping chunk descriptions – even though in the methodbase itself these would be stored as individual non-overlapping fragments. Such duplication, across several chunks, could thus lead to both degradation of quality of the usage of the methodbase overall and to a maintenance problem analogous to the reuse issue that object technology originally sought to remove although. As noted above, this would be ameliorated to some degree at the implementation level since database technology can be used to ensure that only one copy of a fragment exists physically in the repository or methodbase (i.e. storage needs to be "by reference" and not "by value"). The "downside" of this single copy methodbase is that chunks have to be realized outside of the methodbase e.g. in a methodbase chunk catalogue. Such a catalogue is of course unnecessary for the fragment approach [Henderson-Sellers 07b]

Since chunks can be at any granularity (see also [Rolland 96]), it is argued that the full method itself can also be regarded as a chunk (similar to the model adopted more recently in SPEM [OMG 02] in which a Process is modelled as a special kind of ProcessComponent) – see also earlier discussion of Figure 8 in Section 4.1. This type of definition allows not only the construction of methods by assembling chunks but also the tailoring of existing methods as a result of their modularity.

However, while this could work for fragments, since, by the normal definition, a chunk is *one* process-focussed fragment plus *one* product-focussed fragment, then a full software engineering process (SEP) cannot be envisaged as being both a fragment itself *and* being composed of fragments. In other words, there is no meaningful way to model a full SEP as 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 [Henderson-Sellers 07b].

Over the last few years[8] the chunk definition has matured and been revised to take into account more realistic situations of a single process part outputting one or more product parts and also allowing multiple product part inputs e.g. [Ralyté 04a].

Finally, in this approach, associated with every method chunk there is a *descriptor* = <*reuse situation, reuse intention*>, which extends the contextual view captured in the signature or interface, to define the context in which the chunk can be reused. The descriptor also captures additional information such as the origin of the chunk, its type (simple or complex) and a textual description of its objective – see [Rolland 98b, Ralyté 99, Ralyté 01b, Ralyté 01c, Mirbel 06b] for more information on this topic.

In contrast, the fragments generated in the OPEN Process Framework e.g. [Firesmith 02] are all instantiated from a single specific class in the metamodel and

[7] For instance, a pair of overlapping chunks (with a fragment in common) might appear to be a solution but introduces significant and difficult new problems from a conceptual viewpoint.

[8] Also during the preparation of this review in terms of discussions between the two authors.

are weighted towards specifying process elements [Ralyté 04a] as are those of [Mirbel 02] in the JECKO framework. Thus the generated "chunk" is either a process-focussed fragment (e.g. a task, a technique) or a product-focussed fragment (the latter the focus of the proposals of [Hruby 00]). Thus the maintenance needed is solely at the chunk/fragment/component level.

[Wistrand 04] instead use the concept of a *method component* defined as a "self-contained part of a system engineering method expressing the process of transforming one or several artefacts into a defined target artefact and the rationale for such a transformation". This has some similarity with the notion of a process in the ISO 12207 standard [ISO/IEC 95], although it excludes the (potentially common) case of multiple outputs from a process element.

[Wistrand 04] offer two views of a method component: internal and external. In their internal view (Figure 13), a method component consists of method elements and their goals – an addition of values associated with goals was made in [Karlsson 06]. A method element abstracts some part of a method, the three parts of which are listed as actions, concepts and notation; a list revised by [Karlsson 04] as process, concepts and notation. Goals represent the value placed on the element by the creator of that element.

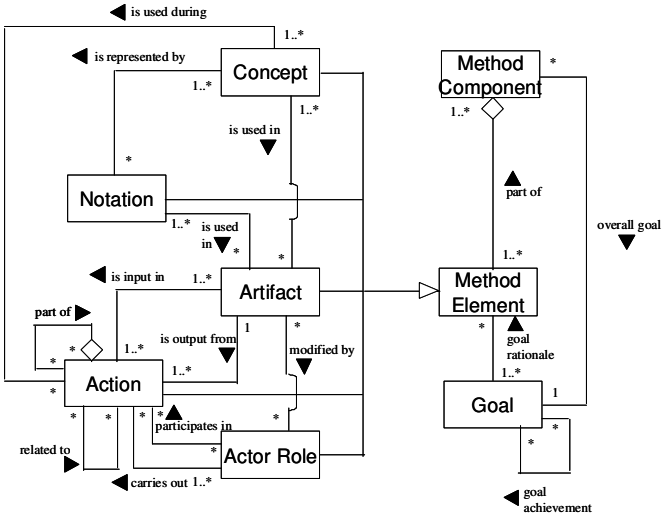

Figure 13: The method component construct (internal view) (from [Wistrand 04]) Copyright Springer-Verlag 2004. Reproduced with kind permission from Springer Science and Business Media

Following the panel discussion at ME'07 [Ågerfalk 07] in which an attempt was made to identify a unique concept for the atomic element for SME, [Deneckère 08] offer, as an alternative to the various fragment/chunk/component definitions discussed above, the notion of a "method service". Although this new and interesting idea needs further research, it too is based on a metamodel, as shown in Figure 14.

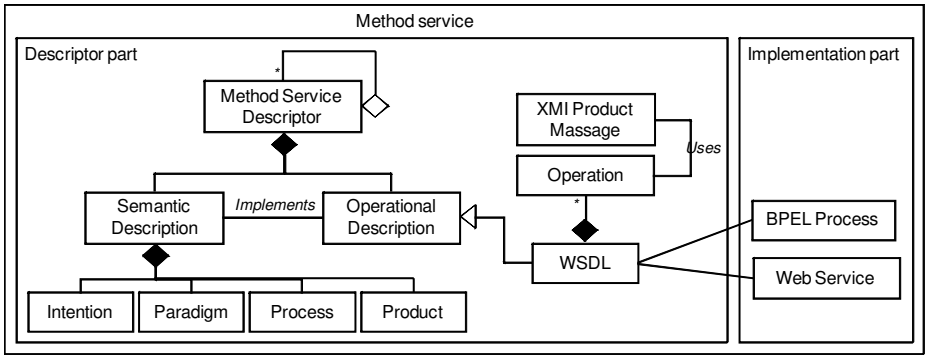

Figure 14: Metamodel for a method service (updated from [Deneckère 08]-revisions kindly supplied by R. Deneckère)

Notwithstanding, since [Ralyté 04a] combines process and product fragments into a single chunk whereas these are kept separate in the OPF, we suggest here that we use a supertype of MethodComponent with subtypes of MethodChunk (for RR usage) and MethodFragment (if using the OPF) or MethodComponent (if using MMC). This is in accord with a recent publication by [Mirbel 06b] in which two further subtypes of Method Component (Pattern and Road-map) are also included.

6 Identification and Construction of Individual Method Chunks/Fragments

The fragment/chunk identification approach proposed in [Ralyté 01b] supports the decomposition of an existing method into method chunks. This is a process-driven approach which considers that it is possible to define the process model of each method in the form of a map. Therefore, as shown in Figure 15, the approach includes four intentions or "steps". The first two intentions represent the process for method map construction as proposed in [Rolland 99] and are relevant when the input method is not available in a modularized format i.e. it has no formalized process model as a map. It consists first of defining map sections, a section being <*source intention, target intention, strategy*> and then defining the associated guidelines (IAG, ISG and SSG). The next two intentions deal with the identification of method chunks from the obtained map by analysing the reusability of the IAG associated with the different map sections and then the definition of method chunks by defining their product parts, interfaces and descriptors.

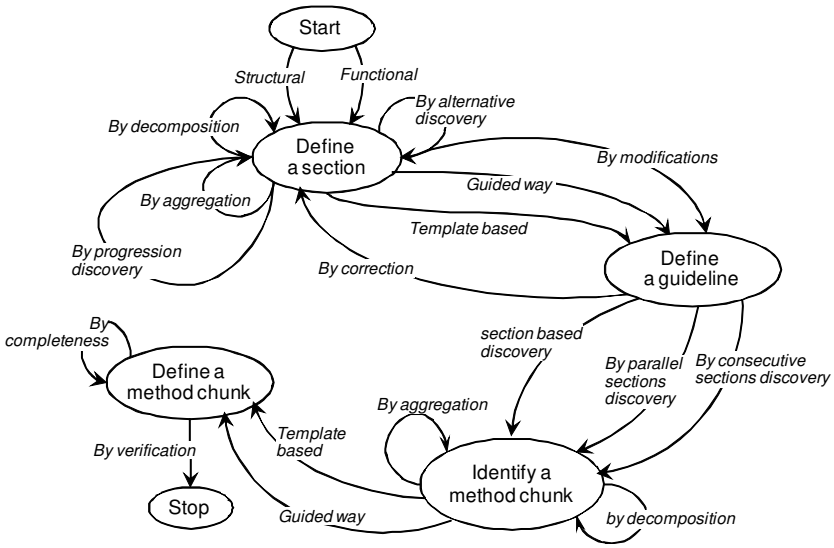

Figure 15: Method reengineering process model. Intentions are shown as nodes (oval symbols) and the strategies by the arcs linking the nodes (after [Ralyté 01b])
Copyright Springer-Verlag 2001. Reproduced with kind permission from Springer Science and Business Media

To satisfy these intentions (of Figure 15), [Ralyté 01b] offer a set of strategies. For example, there are two strategies to satisfy the intention Define a section,: structural and functional. The former is used when there is no pre-existing formal model of the method under consideration, the latter when there is already formal knowledge attached to this method. New sections may then be defined recursively using one of four strategies, as shown in Figure 15. The decomposition discovery and aggregation discovery strategies have obvious meaning; the alternative discovery strategy substitutes a previously existing strategy and the progression discovery strategy helps to create a new section as a progression from an existing one. Strategies associated with the next section (Define a guideline) are threefold: template-based, guided and modification, the last of these linking together changes in guidelines and changes in sectioning. For Identify a method chunk, there are three suggested strategies. The first (section-based discovery) assumes that every section of the map may be regarded as a method chunk with an associated IAG – actually the IAG associated with this section forms the basis for the chunk. The parallel section discovery strategy identifies the IAG associated with parallel sections and aggregates them into a new guideline. Similarly, the consecutive sections discovery strategy assists in identifying the IAG associated with consecutive map sections and integrating them appropriately to give the guideline for the new aggregate chunk. Finally, Define a method chunk has two strategies, similar in character to those for Define a guideline. Following method chunk definition, a verification strategy is necessary in order to ensure that the chunk meets all quality requirements.

[Ralyté 04a] notes that several authors e.g. [Brinkkemper 98, Harmsen 97, Ralyté 01b] propose extracting fragments from existing methods but offer no advice on how to do this. This detailed advice is, however, found in [Ralyté 04a] who describes two approaches: (i) existing method re-engineering and (ii) ad-hoc construction, consistent with the various map sections of Figure 15. These two approaches are described in Sections 6.1 and 6.2 below, respectively. Both are described by the one process (map) model[9] – see Figure 16, which shows two intentions: Identify method chunk and Define method chunk. For the Identify method chunk intention there are four possible strategies: Process-driven decomposition, Product-driven decomposition, Exploration and Ad hoc (thus refining the strategies suggested in [Ralyté 01b] – Figure 15). For the Define method chunk intention there are three possible strategies identified: By completing process part, By completing product part and From scratch. Strategies relating to ad hoc approaches are discussed in Section 6.2. For each map section, guidelines (IAG, ISG, SSG) can be defined – these are discussed in detail in tables 1-3 of [Ralyté 04a].

6.1 From existing methods

Most of the ME and SME literature focusses on identifying and using method fragments/chunks from *existing* methodologies in a plug-and-play format. This is called "existing method re-engineering" by [Ralyté 04a], who notes that since such methods are inherently *non*-modular, extracting appropriate modularized method chunks can present difficulties. To do this, she advocates the use of decomposition or exploration. The former modularizes the whole method and the latter looks at different ways of using the same model where the decomposition could be either process-driven or product-driven. This leads to three of the four strategies shown in Figure 16 for the first section of the process model. She argues that process-driven decomposition is the more likely and more powerful, thus concentrating on the process-driven decomposition and exploration strategies only.

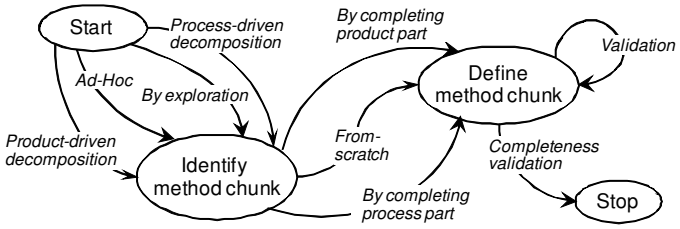

Figure 16: Process model for method chunk construction (after [Ralyté 04a])

The process-driven decomposition corresponds to the first three intentions (and related strategies) of the method re-engineering process model presented in Figure 15.

The second important strategy, according to [Ralyté 04a], is the exploration strategy for method chunk identification. This aims to discover multiple purposes for

[9] From Section 4, we note that a map consists of a number of *intentions* (source and target) linked by a number of (possibly only one) *strategies* (see also Rolland *et al.* [1999]).

an existing model, often when the process part of the chunk is unspecified. This leads to the creation of alternatives for the process model as well as accompanying guidelines. The relevant guideline (IAG3 of table 1 of [Ralyté 04a] states that for the section <Start, Identify method chunk, Exploration strategy>, "This guideline proposes to consider different possibilities to use a model (or a method). The chunk identification process can be based on the goal and/or situation analysis; the same model can be used to satisfy different engineering goals and can be applied in different engineering situations." This is depicted as a map in Figure 17, which shows a single intention, achieved through either a situation-driven or goal-driven strategy.

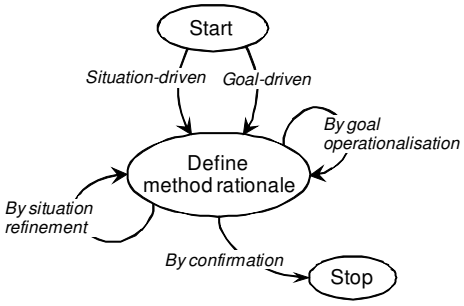

Figure 17: IAG3: Method chunks identification following the exploration strategy of Figure 16 (after [Ralyté 04a])

6.2 From scratch

[Ralyté 04a] discusses constructing new method chunks "from scratch" (as opposed to dissecting existing methods). This is the Ad-hoc approach shown in Figure 16, particularly useful for supporting the emergence of new technologies, such as web development e.g. [Henderson-Sellers 02b] or agents e.g. [Henderson-Sellers 05a]. In this case, theory plus best practice permits the initial identification of fragments or chunks. These are then evaluated in practice, refined, quality assessed in an iterative fashion until quality standards are met. They can then be added to the methodbase although, of course, further iterative refinement in the light of new empirical data, new computing paradigms or new application case studies is to be encouraged. Indeed, the very earliest fragments added to the methodbase for approaches such as that using the OPF were created iteratively in just this fashion.

6.3 Identifying reusable and useful fragments for storage in repository

[Ter Hofstede 97] note that suitability of the fragments for adding to a methodbase requires appropriate coherency and granularity. If the fragment is too coarse, it will require further decomposition after retrieval and before use; if too fine, then more construction effort will be required.

To facilitate later retrieval, it is also important that the fragments suitable for storage are documented succinctly and accurately. [Freeman 91] suggest a structured template (in their case for class interfaces), which can then be searched using their proposed OLMS tool. A similar formalism, targetted towards SME, is provided in [Brinkkemper 01]'s Method Engineering Language (MEL). This formally documents

both product and process fragments, using a number of reserved words in a textual template (e.g. Figure 18). As well as being useful for documenting the fragments themselves, MEL can also be used for the insertion of method fragments into the database using its in-built *methods administration function*.

PRODUCT Object:
LAYER Concept;
PART OF Use Case Model;
SYMBOL Rectangle;
NAME TEXT;
ASSOCIATED WITH {(send,),(receive,)}.

Figure 18: Exemplar description of a product fragment using MEL (as proposed in [Brinkkemper 01])

A similar result ensues from the use of the ISO/IEC 24744 metamodel. For example, the definition of each class in the metamodel follows a format that includes formal specification of items such as:

- Name,
- Attributes,
- Part of,
- Has part,
- Associated with,
- Has superclass.

thus formally defining the ontological characteristics of each metaclass, such as ModelUnitKind (which is used to represent method fragments in the current context).

7 Process for Creating a Methodology from the Fragments

[Ralyté 01a] describe their "modular method meta-model", citing sources of [Rolland 98b, Ralyté 99], as providing the ability to represent any method by an assembly of (reusable) method chunks wherein

- a method is a kind of chunk (at the highest level),
- a method is composed of several chunks,
- a chunk is a kind of guideline (Section 4.2).

A generic process model [Ralyté 03] describes the process by which a software development method(ology) is created as the output of an SME approach. This process model includes three kinds of SME approaches namely Assembly-based, Extension-based and Paradigm-based and permits the combination of them in a particular SME process.

The map describing this process model is shown in Figure 19. It has two core intentions: *Set Method Engineering Goal* and *Construct a Method*. The first of these intentions represents two possibilities for the method engineer: either (1) taking an existing methodology as his/her base and making some adaptations and modifications. These adjustments in this so-called "Method-based strategy" may be in terms of (i)

enhancement, (ii) extension or (iii) restriction. Alternatively, (2) if no base methodology seems appropriate, then a "from scratch" strategy is recommended.

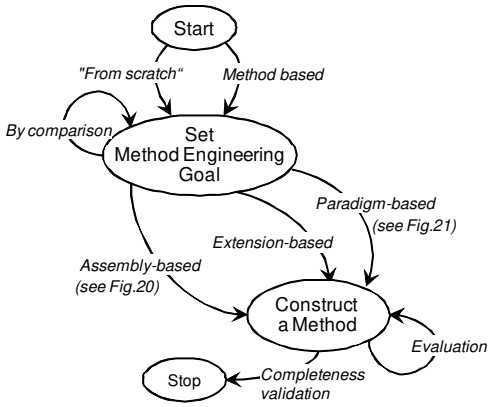

Figure 19: Generic process for SME (after [Ralyté 03]) Copyright Springer-Verlag 2003. Reproduced with kind permission from Springer Science and Business Media

The second core intention in Figure 19, *Construct a method*, can be achieved with one of three different strategies. Using repository stored method components is labelled as an Assembly-based strategy (originally described in [Ralyté 01a] and extended in [Kornyshova 07]). The second option uses patterns applied to existing methods (originally described in [Deneckere 98, Deneckère 01]). The third option, Paradigm-based, may be abstraction-based [Rolland 02, Ralyté 05] or instantiated from a metamodel or adapted [Tolvanen 98]. These three strategies are augmented by a fourth, ad-hoc, by [Ralyté 04b].

Once constructed, the method is then evaluated (Evaluation strategy in Figure 19).

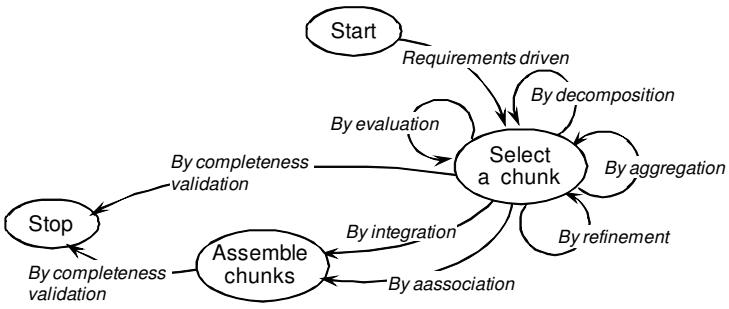

Figure 20: Assembly-based process model for SME (redrawn from [Ralyté 01a, Ralyté 03])

Perhaps of most interest here are the Assembly-based and Paradigm-based strategies. Each section <*Set ME goal, Construct a method, strategy*> from Figure 19 can be expanded, as depicted in Figure 12, for the Assembly-based strategy as shown in Figure 20 and for the Paradigm-based strategy as shown in Figure 21. The third

strategy, Extension-based, is discussed in Section 8 (Figure 30). For the Assembly-based process model (Figure 20), the intention *Select method chunk* is described below in Section 7.2. In turn, the *Requirements-driven* strategy can be expanded [Ralyté 02] as described in Section 7.1. (The Paradigm-based process model (Figure 21) is discussed further below.)

This overall approach is not dissimilar from the contingency based approach of [van Slooten 96] that introduces the notion of a "route map" – a technique used later by [Aydin 02]. In this proposal, there are two kinds of building blocks for SME: the method fragments themselves (of two kinds: standard and adapted) plus route map fragments. The latter may refer to strategies (as in the map idea above), activities and products as well as project management.

[Ralyté 03] carefully distinguish between an Assembly-based approach where the fragments are already assumed to exist, probably having been abstracted from existing methodologies, and a Paradigm-based approach, which generates fragments from a metamodel. Despite their assertion that the key issue to SME is the use of metamodels, this is only rationalized using process maps as one of three possible approaches. This is in contrast to, for example, the work of the OPEN Consortium (described in brief below), in which the whole approach is presaged upon the use of an underpinning metamodel. There, *all* fragments are conformant to the metamodel and stored in the methodbase e.g. [Firesmith 02]. As new technologies are introduced, it is likely that the pre-existing fragments in the methodbase will be insufficient to construct a methodology for the new situation. In such a case, fragments will need to be created *ad hoc* in conformance with the metamodel and then either used for only the new situation or, with appropriate quality control, stored in the methodbase e.g. [Han 08].

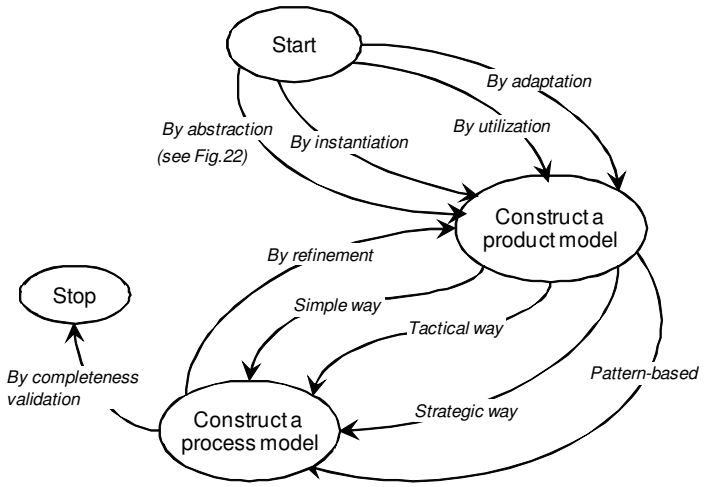

Figure 21: Paradigm-based process model for SME (after [Ralyté 03]) Copyright Springer-Verlag 2003. Reproduced with kind permission from Springer Science and Business Media

The process map for Paradigm-based strategy for method construction, shown in Figure 21, has two main Intentions: Construct a process model and Construct a product model. In [Ralyté 03]'s (and earlier papers), the elemental units of a process/method are taken to be chunks (rather than fragments). Consequently, in Figure 21 we see the necessity (in their approach) to define both the process and product aspects – hence the duality of the intentions. Four strategies are shown to help achieve the Intention: Construct a product model. These four offer the following support:

- Adaptation of a metamodel to specific circumstances,
- Adaptation of a process or product model,
- Instantiation of a metamodel,
- Abstraction by raising or lowering the abstraction level of a given model.

Details of this last strategy are given as an exemplar in [Ralyté 03] – see Figure 22. Although we will not describe the details of this map it is important to see how this is related to maps at higher levels of abstraction. Figure 23 shows how the Abstraction strategy of Figure 22 is embedded within the paradigm-based process model of Figure 21 which in turn is embedded within the generic process map of Figure 19. Although not shown in Figure 23, there is also a downward expansion of each of the strategies shown into their own individual maps i.e. maps for Completeness strategy (indicated), Decomposition strategy, Aggregation strategy etc., with a simple or tactical guideline at the lowest level.

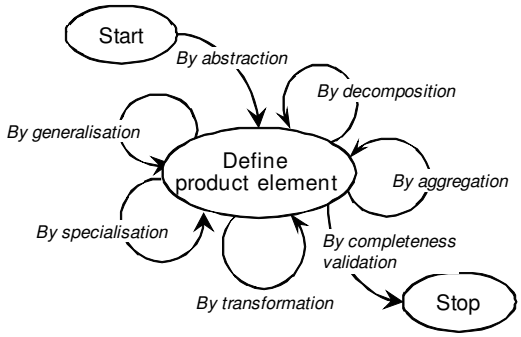

Figure 22: Details of the Abstraction strategy of Figure 21 (after [Ralyté 03])
Copyright Springer-Verlag 2003. Reproduced with kind permission from Springer Science and Business Media)

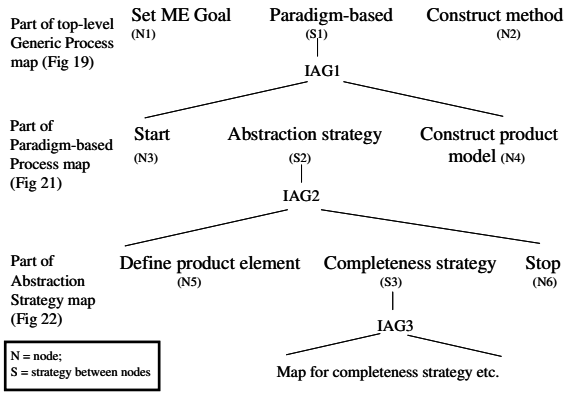

Figure 23: An example of levelling using the template of Figure 12 applied to a part of the Generic Process map of Figure 19 expanded into lower levels of the paradigm-based process (Figure 21) and the abstraction strategy (Figure 22)

For the second Intention in the Paradigm-based process model of Figure 21, namely the Intention: Construct a process model, the appropriate strategies are identified as Simple, Context-driven, Strategy-driven and Pattern-driven with a recursive Refinement strategy. The first two strategies are derived from the NATURE project e.g. [Rolland 95, Jarke 99], the third from [Rolland 99], where it is called "map formalism" and the last proposes to define a process model as a collection of situational patterns. It is noted that this Intention must follow that of *Construct a product model* since the process must conform to the product model. This adds a specific bias to the construction of a method chunk.

A third approach to method construction is the use of deontic matrices introduced in MOSES [Henderson-Sellers 94] and SOMA [Graham 95b] and now embodied in the OPEN Process Framework [Graham 97, Henderson-Sellers 98b, Firesmith 02 – see also http://www.open.org.au] [Henderson-Sellers 04b] and captured and formalized with the DeonticValue enumerated type in ISO/IEC 24744. A deontic matrix (Figure 24) is a two dimensional matrix of values that represent the possible or likely relationship between each pair of method fragments chosen from the OPF repository. Such a matrix can be used to assist in the construction of a methodology. For example, for each activity (column A may represent the Requirements Engineering activity for instance), the matrix suggests that for this activity (column labelled A), the use of Task 1 is Mandatory, Tasks 2 and 3 are unlikely to be useful and the inclusion of Task 10 is solely optional for the project team to choose or reject. (Other levels of possibility are indicated in the key of Figure 24 and in more detail in [Graham 97, Henderson-Sellers 02a]).

[Nguyen 03] identify seven possible deontic matrices: (1) Process/Activity, (2) Activity/Task, (3) Task/Technique, (4) Producer/Task, (5) Task/Work Product, (6) Producer/Work Product and (7) Work Product/Language. To determine the values in these several matrices, there are a number of factors such as project size, organizational culture, domain of the application to be developed, and the skills and

preferences of the development team that influence the method engineering in their evaluation of the most appropriate values for the deontic matrices (Table 1). Once completed, these matrices give guidance on the most appropriate selection of method fragments. As experiential data are gathered, a knowledge database is built up, which can then provide a "first guess" (via an automated tool) on which the method engineering can then elaborate for the very specific situation [Nguyen 03]. With an automated tool to help, not only can the method engineer be helped to identify appropriate method fragments, but web-based documentation can be automatically produced [Henderson-Sellers 04a].

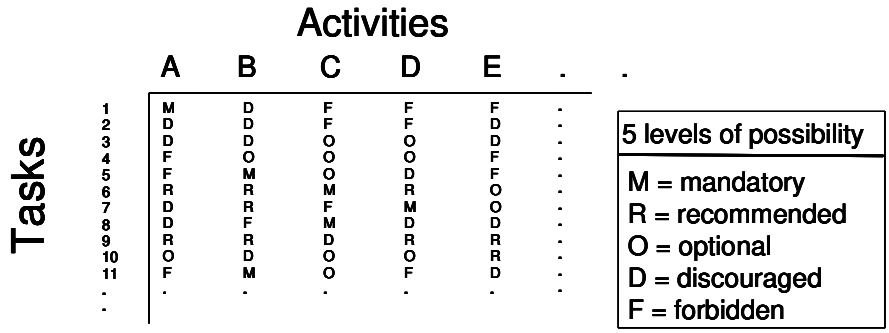

Figure 24: A core element of OPEN is a two-dimensional relationship between each pair of methodology elements, here shown for Activities and Tasks. Each Activity may require one or more Tasks. For each combination of Activity and Task, an assessment can be made of the likelihood of the occurrence of that combination using five levels of possibility (labelled M, R, O, D or F) (after [Firesmith 02]). Reproduced with permission from Pearson Education Ltd.

The series of selections (i.e. usage of each of these seven matrices) can be variable, as determined by the method engineer. Some prefer to start top down, by first identifying Activities and then asking what Tasks are needed for those Activities and then what Techniques might be useful to implement the selected Tasks. A typical exemplar is to be found in [Henderson-Sellers 04b]

Other method engineers prefer to start with lower level elements, perhaps starting with tasks or with work products [Gonzalez-Perez 08a]. Whichever approach (top-down or bottom-up) is chosen, a set of deontic matrix values is the end result. These are highly dependent upon the method engineer's decisions, based on the organizational response to questions such as those in Table 1. A comparison of two such matrices, one for a small B2C project and one for a large B2C project, derived from two in-depth industry case studies [Haire 00], is presented by [Henderson-Sellers 02a].

Question	Possible Answers
What is the maturity level (by CMM standard), at which your organization is currently assessed or aims to achieve?	Level 1 (Initial). Level 2 (Repeatable). Level 3 (Defined). Level 4 (Managed). Level 5 (Optimizing).
Which of the following best describes the domain of the software product being developed?	E-Business Embedded system MIS Process control Real time
What is the level of quality that the information system should have?	Low, Normal, High.
What is the estimated size of the final software product being developed?	Small and not complex (less than 10,000 LOC). Medium and moderately complex (between 10,000 LOC and 50,000 LOC). Large and very complex (between 50,000 LOC and 1,000,000 LOC). Very large and immensely complex (more than 1,000,000 LOC).
What is the estimated size of the project team on this project?	Small (1-3 members). Medium (4-9 members). Large (more than 9 members).
What is the level of criticality of the software product being developed to a successful mission?	Low, Normal, High, or Very high
What type of user interface is required?	No user interface involved. Text based user interface. Graphical user interface.
To what extent does the project depend on activities from other projects within the organization?	Low, Normal, High.
To what extent do the members of the project team possess enough knowledge and experience to develop the required software product?	Low, Normal, High.
To what extent will the goals, needs, and desires of the users remain stable over time, thus enabling a stable specification of the functional requirements to be made?	Low, Normal, High.
To what extent are the goals, needs, and desires of the users clear and coherent, thus enabling a sound specification of the functional requirements to be made?	Low, Normal, High.
To what extent is there sufficient time available for the project?	Low, Normal, High.
To what extent was the applied technology and/or the applied methods, techniques and tools new to the organization?	Low, Normal, High.
To what extent is the level of reuse required in the development project?	Low, Normal, High.
Does the software product being developed involve the use of a database system?	Does not involve the use of a database. Use an OO database. Use a non-OO database.
Does the software product being developed involve a distributed environment?	Yes or No

Table 1: Questions and answers for incorporation into a tool for automated generation of deontic matrices (after [Nguyen 03])

With the advent of the ISO/IEC International Standard 24744 [ISO/IEC 07], there are new possibilities for the "glue" between fragments since this metamodel has a number of metaclasses specifically designed to link the major method fragments together. These include ActionKind and WorkPerformanceKind.

In addition, increasingly sophisticated teams will wish to see their process mature commensurately. Using the deontic matrix approach, new method fragments are easily added to an organization's existing methodology. Thus, the SME approach also provides implicit support for SPI (Software Process Improvement) [Henderson-Sellers 06b, Henderson-Sellers 07a, Bajec 07b] and ISO/IEC 24744 has an explicit metaclass to represent the capability level that will be used in such a software maturity evaluation (using, say, ISO 15504: [ISO/IEC 98]).

A fourth possibility [Seidita 07] is the relatively straightforward application of UML Activity Diagrams (ADs). However, ADs are most useful for depicting the workflow of the process model once the fragments have been both selected and put together and seem to offer less support in fragment selection and assembly, wherein the method engineer's expertise plays a more important part. Although [Van de Weerd 06, Saeki 03] use a variation on this approach in which they link an AD to an architectural style model to depict the associated product part, this too is largely a descriptive (rather than prescriptive) approach to process modelling, relying again on the method engineer being able to select a set of existing method fragments that could fit the application context on the basis of personal knowledge and expertise.

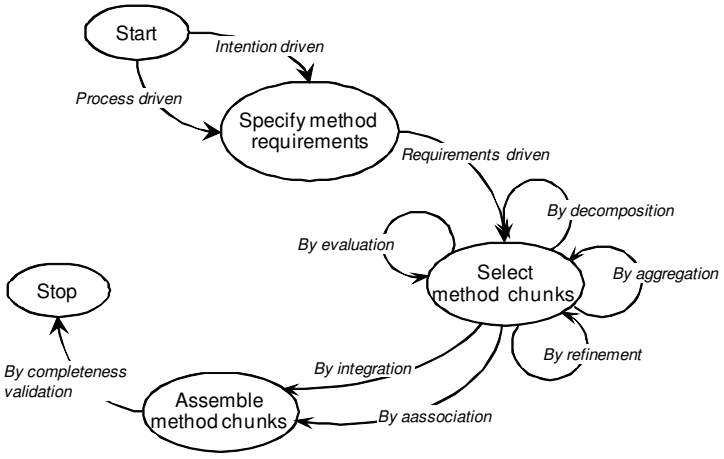

Figure 25: Assembly-driven process model including explicit description of requirements specification (as a new intention plus two new strategies) (after [Ralyté 03]) Copyright Springer-Verlag 2003. Reproduced with kind permission from Springer Science and Business Media

7.1 Requirements of the intended method

Identifying the requirements of the to-be-constructed method involve all members of the development team and the management team(s). This assumes that it is feasible that these people can identify the kind of method and its characteristics that will suit

their forthcoming project and that the various perspectives can be integrated [Nuseibeh 96] and then communicated to the method engineer [Karlsson 05]. It is also typically assumed that these requirements do not change during the project. (Changing requirements are considered by e.g. [Rossi 04] in their study of evolutionary method engineering.) This, of course, has many similarities with traditional requirements engineering in the context of software development, rather than method construction.

The Requirements-driven strategy shown in Figure 20 considers that method requirements have been defined previously to the method chunks selection and assembly. The assembly-driven process model, described by a map in Figure 25, explicitly introduces the intention Specify Method Requirements as a first step in the situation-specific method construction and proposes two possible strategies by which to achieve this intention (Intention-driven strategy and Process-driven strategy) [Ralyté 03].

In the process-driven strategy, applicable to the situation when the method engineer needs to construct a completely new method, a form of map called a requirements map is produced. This strategy applies the Map construction guidelines proposed in [Rolland 99], which were also adapted in the method chunk construction process [Ralyté 01a, Ralyté 04a] (see Section 7, Figure 15). The requirements map construction process is limited to the identification of its sections without associating the corresponding guidelines. In fact, the guidelines will be associated to this map during the method chunk selection process. The first set of sections is identified by using the Activity-driven strategy, which is based on the analysis and selection of the engineering activities to be supported by the method. Then, this collection of sections can be refined by decomposing or aggregating already identified sections or by discovering alternative or complementary sections. Finally, the Verification strategy helps to finalize and check the utility of the proposed requirements map. [Ralyté 02] shows by example how this map is used to create a requirements map. The example chosen is that of constructing a method supporting the analysis and design of a B2B system. This follows the elaboration of each strategy and guidelines as a map in a recursive fashion (parallel to Figures 23 and 12). The final requirements map (Figure 26) is then used as the basis for the method chunk selection and assembly process depicted in Figures 19 and 25.

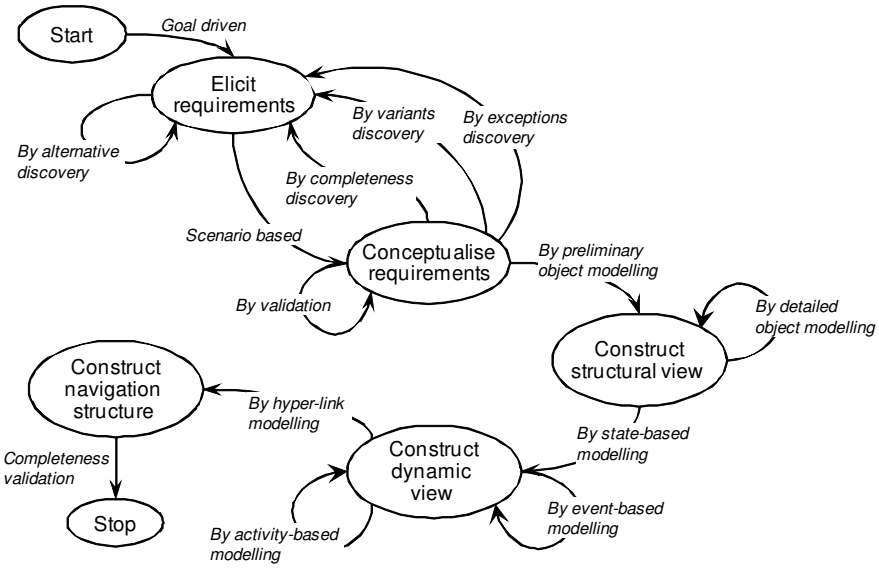

Figure 26: Final requirements map (after [Ralyté 02]) Copyright Springer-Verlag 2002. Reproduced with kind permission from Springer Science and Business Media

7.2 Identifying useful fragments that exist in the repository

In this top down method construction approach, identification of useful fragments is the remit of the intention Select method chunks in the assembly-based process model of Figure 20. For each retrieved chunk, its potential usefulness is first evaluated (evaluation strategy) – this can be done using similarity measures as described by [Ralyté 01a] and extended by [Mirbel 06a], who describes three kinds of similarity: (1) the number of common aspects based on "User Situation" and "Reuse Context", (2) the forbidden aspects of "User Situation" and "Reuse Context" and (3) the number of necessary aspects in the "User Situation".

When necessary, refinement of the chunk may be undertaken using one of three further strategies [Ralyté 01a]:

- decomposition strategy – where the chunk is a compound one containing parts not needed for the current method construction,
- aggregation strategy – when the chunk only covers the requirements partially,
- refinement strategy – suggests seeking another chunk with a richer set of guidelines than the current selection.

The meta-knowledge stored with the method fragment is highly relevant in ensuring a contextual retrieval. Suggestions for an appropriate query language are given in [Rolland 96]. The modelling language, MEL, proposed by [Brinkkemper 01] also contains a portion useful for identifying and removing method fragments from the database.

7.3 Quality aspects of the constructed method

While [Tolvanen 96] raised the issue of how to standardize the quality of engineered methods, little research has since been undertaken in this area. The created method clearly needs to be validated/of high quality. [Chroust 00] discusses the well-formedness of process models, while [Brinkkemper 98] stress the need to assemble a *meaningful* method from the retrieved fragments by, for example, using two fragments at the same granularity. It is not usually appropriate to assemble a method for industrial usage with a class diagram and an ER diagram because of the extensive overlap between these two fragments, although there are some situations (e.g. in a research mode, creating a method for doing parallel modelling of the same information using class diagrams and ER diagrams, and then exploring the commonalities and differences or for purposes of teaching the differences between the two approaches).

[Brinkkemper 98] focus on possible defects that might occur as a result of the assembly process. These include:

- Internal incompleteness. There is a reference to a second fragment that has not been included in the constructed method;
- Inconsistency. Contradictions[10] between a pair of selections – for example, the selection of two similar techniques to fulfil one particular task without due consideration or rationale being given i.e. thoughtless selection of highly similar method fragments;
- Inapplicability. Selected fragments cannot be applied by project team members due, usually, to insufficient capability.

These defects may occur in the context of the internal or situation-independent quality [van der Hoef 95] for which the most important criteria are completeness and consistency (others being efficiency, reliability and applicability: [Harmsen 97]).

However, [Nuseibeh 96] note that full consistency is not generally achievable. They discuss means of providing partial consistency and provide some suggestions for rules for consistency management. In some contrast, [Brinkkemper 98] note the obvious ease with which meaningless constructions of "methods" can be made by unthinking combination of fragments.

[Brinkkemper 98] propose twelve rules (Table 2) to ensure that the constructed methodology is of high quality. Rules 1-6 refer to method fragments in the conceptual layer and the diagram layer. Rule 7 relates to the diagram layer and Rules 9-11 with the conceptual modelling fragments. [It should be noted that (i) these rules assume that a process element acts merely as a transformation engine i.e. it has of necessity one input and one output and (ii) Rule 9 has some exceptions – some work products, especially the first work product input to the first process element, may be supplied externally, having been created outside the software development environment and used as a "seed input" to initiate the method].

[10] Brinkkemper *et al.*'s use of the word contradiction here is perhaps too strong since inconsistencies reflect sub-optimal selection rather than actual contradictions.

1.	There should be at least one concept newly introduced in each method fragment.
2.	There should be at least one concept linking the two fragments to be assembled.
3.	When adding new concepts, there should be connections between them and existing fragments.
4.	When adding new associations, both new fragments should be participants.
5.	In the resultant combined fragment, there should be no isolated elements.
6.	There should be no name duplication for different method fragments.
7.	Identification of added concepts should occur after the associated concepts have been identified.
8.	When two fragments are assembled, it is necessary that the output of one is used as the input to the other.
9.	Every work product must be identifiable as the output of a particular process fragment.
10.	When a work product has been created from other work products, then the process fragments producing the individual work products are summed to the process producin the amalgmated work product.
11.	Any technical method fragment should be supported by a conceptual method fragment.
12.	When there is an association between two product fragments, there should be at least one association between their respective components.

Table 2: Rules proposed in [Brinkkemper98] for ensuring constructed process quality

[Hassine 04] also use the map technique as a means of discussing how to incorporate quality fully into a method engineering approach. Figure 27 depicts the various intentions and strategies to support software quality assurance (SQA) using the three standard guidelines discussed above (IAG, ISG, SSG). These guidelines are used to decompose the SQA process into sub-processes. [Hassine 04] then underpin this with a metamodel based, inter alia, upon that of [Ralyté 01a]. They then use this situation-specific SQA approach to create an exemplar, called WinWin, which identifies and provides a negotiation framework for the resolution of conflicts between the quality requirements.

More recently, [Han 08] have proposed tests for quality assessment including metrics such as domain coverage (whether the domain fragments, together, provide adequate coverage of the domain) and access paths (whether the repository provides sufficient access paths to reach a given domain fragment); but note that the decisive measure of quality will be whether systems designers use the fragments in practice and whether the resultant methodology is superior to one that is purchased "off the shelf".

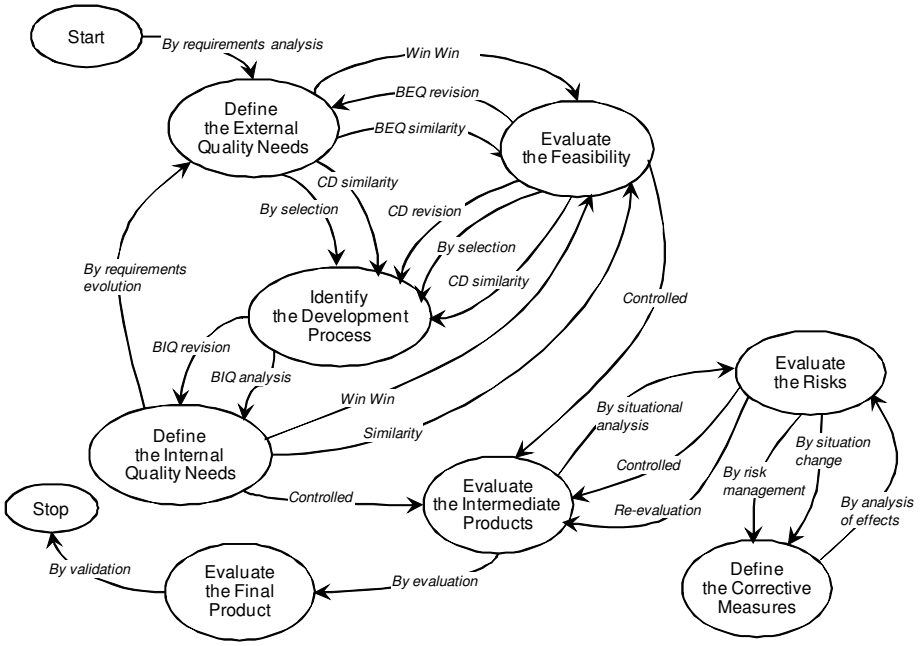

Figure 27: Map for SQA (modified from [Hassine 04])

8 Tailoring a constructed method

[Fitzgerald 03] note a recent recognition that "off-the-shelf" methods need to be tailored to fit the needs of a specific project, even if the method appears to be appropriate and suitable for the project in hand [Aydin 02]. [Fitzgerald 03] focus on the usefulness of (a) contingency factors and (b) method engineering and show how this was successful within a Motorola case study. [Kokol 99] argues that the failure of IT in the medical area can be attributed to the inappropriateness of the methodology used – offering method engineering as a remedy. [Arni-Bloch 06] show how a situational method engineering approach to the integration of COTS packages into more traditional information systems can be efficacious. [Henderson-Sellers 05b] have undertaken an extensive action research study of several organizations who chose to create an agile method from method fragments. The resulting method of one of those organizations is depicted in Figure 28 where each ellipse represents a high level Work Unit (called Activity in the OPF) and connecting lines embody the chosen links based on a contract-driven lifecycle model [Graham 95a]. Industry experience leads [Bajec 07a] to suggest Process Configuration as an effective tailoring approach to create situation-specific methods all based on a single base approach, which is itself allowed to continuously evolve.

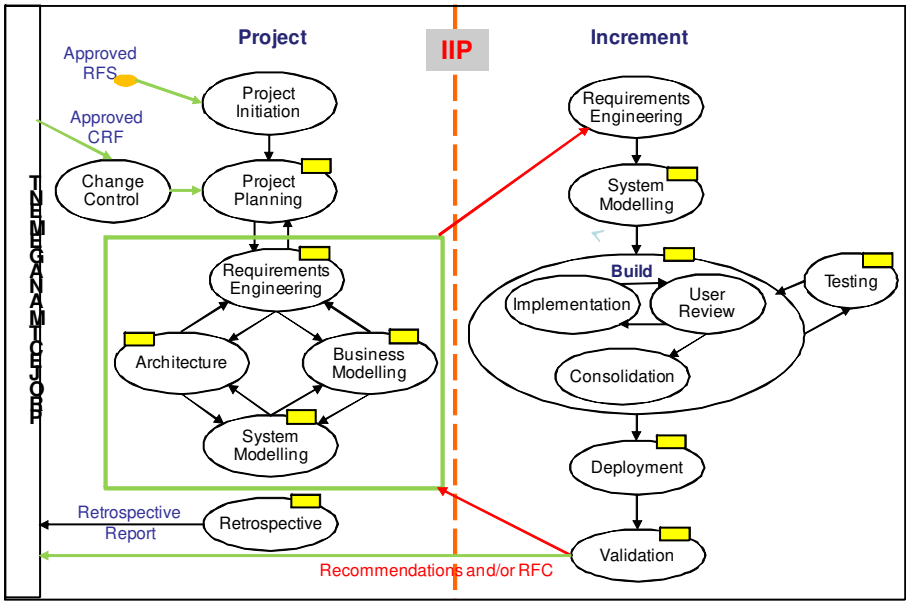

Figure 28: The engineered method for the study organization (after [Henderson-Sellers 05b])

The Motorola case study also provides an excellent depiction of how process tailoring works in practice. [Fitzgerald 03] identified two types of tailoring at the project level: up front (recorded as part of the project plan) and dynamic (more ad hoc in light of modified perceptions of the project as it proceeds). In both cases, they found that that capture and definition of tailoring criteria were essential in risk management and in terms of the overall success of the project. In the context of manufacturing processes, [Rupprecht 00] discuss tailoring as "process individualization" i.e. adapting a process model for a specific context. [Patel 04] use, as an exemplar for their tailoring approach, the Rational Unified Process [Kruchten 99, MacIsaac 03]; whereas [Aydin 02, Aydin 05] utilize as their base methodology the DSDM (Dynamic Systems Development Method: [Stapleton 97]). They undertook an empirical assessment with one specific large financial organization, showing disparate levels of understanding across the management team. They instigated a set of coaching activities to assist in dissemination of this information and understanding of tailoring, in the context of aiming to achieve CMM level 3. The process that resulted is shown in Figure 29.

There are also situations when a base method is kept but various versions of it are supported – as a "family" of methods. This has happened in Germany with the V-Modell XT [Rausch 05] and, in the solely product domain, with the UML family of modelling languages as specified by the numerous "profiles" that now exist. In both cases, there exists the challenge of managing the variability of a family of tailored approaches (plus the "root" one) as they are used and applied.

[Wistrand 04] discuss method configuration (MC) as a special case of ME. For them, MC has three strategies:

1. selecting parts from an assumed pre-existing base method,
2. integrating parts from other methods to fill gaps in base method,
3. developing new parts when (b) not applicable,

[although it should be noted that this appears not to support restriction as they state!].

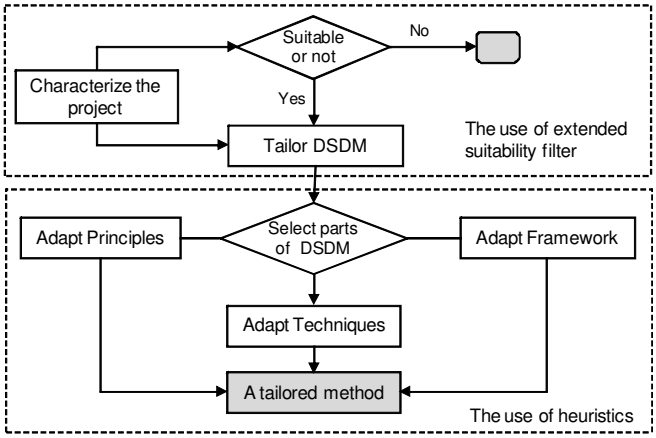

Figure 29: Overall coaching activities used in method tailoring in an empirical study (after [Aydin 02]). Copyright Springer-Verlag 2002. Reproduced with kind permission from Springer Science and Business Media

[Wistrand 04] state that, of [Ralyté 03]'s three strategies (viz. assembly based; extension based and paradigm based), extension based is nearest to their SEM based MC strategy but (they claim) MC permits not just extension but also restriction (which they state is not possible using the Ralyté *et al.*'s extension strategy). This extension-based strategy can be expanded from Figure 17 as shown in Figure 30. Two strategies are shown: domain-driven in order to select a Meta-pattern followed by a pattern-based strategy or else a pattern-matching strategy.

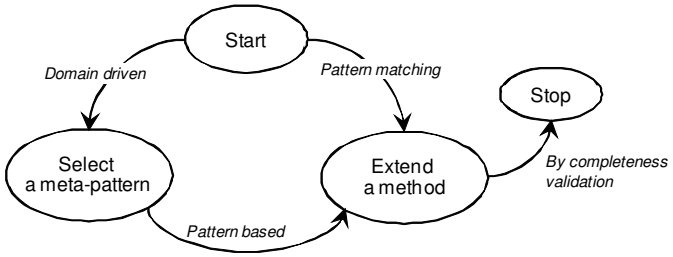

Figure 30: Extension-based process model for SEM (after [Ralyté 03]) Copyright Springer-Verlag 2003. Reproduced with kind permission from Springer Science and Business Media

[Chroust 00] identifies three kinds of process tailoring:

- Reductive. Take a comprehensive model and remove the unwanted pieces.
- Synthetic. Start with fragments and build up the process. Union operators are needed here.
- Generic. This is not clear in the original publication but appears to refer to the use of a fragment as a parameter for the generation of another fragment.

Tailoring usually involves removal of unwanted bits. Therefore useful operators are:

DIFFERENCE ($PM_1 - PM_2$) – returning those elements which are in one method but not the second. Elimination of an activity thus involves the subtraction of a unit set from the method by using this DIFFERENCE operator;

INTERSECTION ($PM_1 \cap PM_2$) – returning the common elements of the two process models

EQUALITY ($PM_1 = PM_2$) – allowing the comparison of two models. The relation is true if $PM_1 - PM_2$ is the empty set. However, it should be noted that the empty set simply states that the method fragments in the two sets are identical. It does not guarantee that the configuration of those two method fragment sets is the same.

Process tailoring and process knowledge are discussed by [Xu 02] who propose a framework (described by a metamodel) as a "first step in supporting the process reuse … to represent the elements needed to be captured in process tailoring". They go on to propose the design for a prototype tool developed to capture and use such process knowledge.

8.1 Process for method configuration

[Karlsson 02] focusses on one special aspect of SME, that of tailoring an existing or base method; and compares this approach with more traditional SME. He describes a method by which such method tailoring or "method configuration" could be accomplished – termed Method for Method Configuration (MMC) [Karlsson 01]. The scope is that of modifications to off-the-shelf software development methods like RUP [Kruchten 99] to produce a situational method. MMC starts with a pre-selected base method, which could be a full method or a set of fragments, existing in some identifiable context. It is then broken down into a sequence of activities to be performed, perhaps grouped into sub-processes.

[Karlsson 02] argues that an essential activity in both MMC and SME should be grounding in order to assure quality, that grounding being performed in three parts: internal, external and empirical. Internal grounding focusses on whether the concepts used are complete and consistent; external grounding relates this knowledge to other, external sources in the same domain. Finally, empirical grounding tests and evaluates the method in everyday use. As a prerequisite to undertaking external grounding, he identifies five essential concepts: Base Method (the method that is used as the basis for the method tailoring); a Development Situation (an abstraction of one or more existing or future projects with common characteristics); Characteristic (a delimited part of a development situation); a Development Track (a method configuration suitable for some specific situation – renamed Configuration Package in [Karlsson 04]); a Generic Project Type (that predefines a combination of Development Tracks in order to facilitate reuse across commonly occurring situations – renamed as Configuration Template in [Karlsson 04] (augmented by Prescribed Action, which represents a process fragment, in [Karlsson 04]). [Karlsson 02] then identifies the

mappings between these five concepts in MMC and SME – summarized in Table 3 and exemplars presented by [Karlsson 05]. More recently [Karlsson 09], the idea of a Configuration Package, which typifies the situational context for the tailoring (in terms of, for instance, co-location of project team, availability of customer(s), project risk level, degree of management commitment), has been applied in an interesting analysis of empirical data from the use of MMC in agile computing domains. These authors conclude that MMC is useful as a quality assurance tool; and that goals can be useful as a reference point when extending the base method.

Concept in MMC	**Mapped concept in SME**
Base method	SME usually (but not always) starts with fragments not a whole method so there is no corresponding concept in SME. The correspondence is to the finally constructed method rather than anything initially.
Development situation	An equivalent concept is found in much of the SME literature.
Characteristic	Although the concept exists in SME, MMC offers a more straightforward link to its effect on the configuration. In a bottom-up approach, the value of a characteristic value is potentially a method fragment. In MMC, however, there is a one-to-one mapping between the characteristic's value and a Development Track.
Development Track	There are large differences in this concept. MMC focusses on screening of the base method rather than modularization, which is the common focus in the SME literature. However, operationalized Development Tracks do have some similarities with, for example, method fragments and components, but with vaguer boundaries.
Generic project type	Templates for systems engineering methods (SEMs) are not a new phenomenon in SME; for example, the use of paths in SEMs.

Table 3: Summary of external grounding (adapted from [Karlsson 02])

[Perez 95] discuss congruence evaluation, arguing that this is the most important measure to be assessed. Congruence assesses how well the process model fits the intended usage/domain. They suggest that, in order to measure congruence, a contingency model must first be developed in order to define the relationships between the process model and its intended usage context. They introduce three variables: a dependent variable (the effectiveness of the process model), an independent variable (a characteristic of the process model) and a contingency variable (either a characteristic of the context or the process model). Attribute values must first be assigned, often subjectively by someone familiar with the situation, and then a congruence measure calculated based on the inter-relationships established between the process model and the attributes of the process context. The derived congruence index is a real number in the closed interval [-1,1] with 1 indicating a

perfect match and -1 the worst possible match. Low congruence values can thus be identified with the aim of improving the underlying values and thus increasing that specific value. A similar, contingency approach is also advocated by [van Slooten 96], based on work of [van der Hoef 95]. Based on a banking example and field trial, [van Slooten 96] recommend the following list of contingency factors:

- Management commitment,
- Importance,
- Impact,
- Resistance and conflict,
- Time pressure,
- Shortage of human resources,
- Shortage of means,
- Formality,
- Knowledge and experience,
- Skills,
- Size,
- Relationships,
- Dependency,
- Clarity,
- Stability,
- Complexity,
- Level of innovation.

Adaptation of processes is also discussed by [Henninger 02] in the context of agile methodologies.

9 Summary and Future Work

Method engineering or, more specifically, situational method engineering (SME) has an extensive but disparate history. We have here drawn together the various strands of the SME literature in the context of formalizing and regularizing the conceptual framework and underpinning theory. Following the conceptualization of the issues involved, expressed by the model of Figure 3, we have described the state-of-the-art with respect to metamodelling for method fragments and chunks, descriptions of methods and chunks themselves and processes by which these can be identified, created and amalgamated into new methods. A significant contribution here comes from the application of the map concept of [Rolland 99] as well as some discussion of alternatives such as the deontic matrix approach. The survey is concluded with an evaluation of some ideas on method tailoring and the emerging research on quality evaluation applied to SME.

However, we do not attempt to undertake a formal, framework-based comparison since the various approaches are still in their infancy in terms of empirical, industry-based data. Initial data (Section 8) are promising but as yet sparse. There also still remain theoretical differences, as seen in this review, for which active collaboration is attempting to resolve (see, for example, the panel discussion at the IFIP conference in September 2007: [Ågerfalk 07]. Thus a combination of theoretical advances coupled

with empirical data will likely advance the state-of-the art in SME in the next few years.

Although we have presented a state-of-the-art as of the time of writing, research continuously moves the field on. The likely topics for research initiatives over the next few years would seem to us to need to include:

- How to best gather the requirements of an organization for their specific situational method;
- How to move from these requirements to a semi-automated way of identifying the optimal collection of fragments (some preliminary work is to be found in [Henderson-Sellers 04a, Garcia-Ojeda 09]);
- How to ensure that the three basic elements (as in Figure 5) are utilized equally (much of the SME literature has ignored the Producer fragments);
- How to ensure that the configuration of these selected method fragments is consistent and complete;
- How to avoid clashes of mindset e.g. using an agile fragment together with a bureaucratic (high ceremony) fragment;
- How to evaluate the quality of the process (in terms of say utility in its enactment and practical support given to the organization);
- How to use method engineering in the context of existing (legacy) methods (most of the SME literature assumes greenfield projects);
- How to model and market "Method as a Service" (MaaS) – an analogue to SaaS [Rolland 09];
- What sort of tools can be created to support industry adoption of an SME approach (some initial ideas, commensurate with ISO/IEC 24744 metamodel, are to be found in [Gonzalez-Perez 05a] and are the focus of a conference workshop in 2010: http://www.enase.org/ISOMeta_CTT.htm).
- The challenges for industry include:
- Full cost calculation of the use of SME versus off-the-shelf approaches;
- Acknowledging the value to the company of "owning" the methodology;
- Assessing the positive value of people-focussed process elements compared to methodological approaches that are imposed externally and possibly seen as "sterile":
- The creation of a new generation of CAME tools build on internationally standardized methodology metamodels.

Acknowledgements

BH-S wishes to acknowledge financial support from the Australian Research Council. JR wishes to acknowledge financial support from the Swiss National Science Foundation under grant N° 200021-103826. We also wish to thank Cesar Gonzalez-Perez, Daniel Pfeiffer and Asif Qumer, who gave us useful comments on earlier drafts of this paper. This is Contribution number 09/02 of the Centre for Object Technology Applications and Research.

References

[Ågerfalk, 04] Ågerfalk, P.J., Karlsson, F.: Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets. Information and Software Technology 46(9), 2004, 619-633.

[Ågerfalk, 07] Ågerfalk, P.J., Brinkkemper, S., Gonzalez-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., Ralyté, J.: Modularization constructs in method engineering: towards common ground? in Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 359-368

[Ambler, 98] Ambler, S.: Process Patterns: Building Large-Scale Systems Using Object Technology. SIGS Press, New York, NY, USA, 1998.

[Ambler, 99] Ambler, S.: More Process Patterns: Building Large-Scale Systems Using Object Technology. SIGS Press, New York, NY, USA, 1999.

[Arni-Bloch, 06] Arni-Bloch, N., Ralyté, J., Léonard, M.: Integrating Information Systems Components: A Situation-Driven Approach. In CAiSE'06. 18th Conference on Advanced Information Systems Engineering – Trusted Information Systems. Luxembourg 5-9 June, 2006. Proceedings of the Workshops and Doctoral Consortium, T. Latour, M. Petit, Eds. Namur University Press, Belgium, 2006, 433-444.

[Atkinson, 98] Atkinson, C.: Supporting and Applying the UML Conceptual Framework. In «UML» 1998: Beyond the Notation, J. Bézivin, P.-A. Muller, Eds. Springer-Verlag, Vol. 1618, Berlin, 1998, 21-36.

[Atkinson, 01] Atkinson, C., Kühne, T.: Processes and Products in a Multi-Level Metamodeling Architecture. International Journal of Software Engineering and Knowledge Engineering 11(6), 2001, 761-783.

[Avison, 91] Avison, D.E., Wood-Harper, A.T.: Information Systems Development Research: An Exploration of Ideas in Practice. The Computer Journal 34(2), 1991, 98-112.

[Avison, 96] Avison, D.E.:Information Systems Development Methodologies: A Broader Perspective. In Method Engineering. Principles of Method Construction and Tool Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA, S. Brinkkemper, K. Lyytinen, R.J. Welke, Eds. Chapman & Hall, London, 1996, 263-277.

[Avison, 03] Avison, D.E., Fitzgerald, G.: Where Now for Development Methodologies? Communications of the ACM 46(1), 2003, 79-82.

[Aydin, 02] Aydin, M.N., Harmsen, F.: Making a Method Work for a Project Situation in the Context of CMM. In Product-Focused Software Process Improvement: 14th International Conference, Profes 2002 Rovaniemi, Finland, December 9-11, 2002, M. Oivo, S. Komi-Sirviö, Eds. Springer-Verlag, LNCS, 2559**,** 2002, 158-171.

[Aydin, 05] Aydin, M.N., Harmsen, F., Van Slooten, K., Stegwee, R.A.: A Model for a Method Adaptation Process. In Information Systems Development. Advances In Theory, Practice and Education, O. Vasilecas, A. Caplinskas, G. Wojtkowski, W. Wojtkowsi, J. Zupanicic, Eds. Kluwer Academic/Springer Publishers, New York, NY, USA, 2005, 477-487.

[Baddoo, 03] Baddoo, N., Hall, T.:De-motivators for Software Process Improvement: An Analysis of Practitioners' Views. Journal of Systems and Software 66, 2003, 23-33.

[Bajec, 07a] Bajec, M., Vavpotič, D., Krisper, M.: Practice-Driven Approach for Creating Project-Specific Software Development Methods. Information and Software Technology 49, 2007, 345-365.

[Bajec, 07b] Bajec, M., Vavpotič, D., Furlan, S., Krisper, M.: Software process improvement based on the method engineering principles, In Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 283-297

[Barbier, 03] Barbier, F., Henderson-Sellers, B., Le Parc-Lacayrelle, A., Bruel, J.-M.: Formalization of the Whole-Part Relationship in The Unified Modeling Language. IEEE Transactions on Software Engineering 29(5), 2003, 459-470.

[Basili, 87] Basili, V.R., Rombach, H.D.: Tailoring the Software Process to Project Goals and Environments. In Proceedings of the Ninth International Conference on Software Engineering, March 30 – April 2 1987, Monterey, CA, 1987.

[Becker, 03] Becker, J., Kugeler, M., Rosemann, M.: Process Management – A Guide for the Design of Business Processes. Springer-Verlag, Berlin, Germany, 2003.

[Bergstra, 85] Bergstra, J., Jonkers, H., Obbink, J.: A Software Development Model for Method Engineering. In Esprit'84: Status Report of Ongoing Work, J. Roukens, J. Renuart, Eds. Elsevier Science Publishers .V., North-Holland, 1985.

[Berki, 04] Berki, E., Georgiadou, E., Holcombe, M.: Requirements Engineering and Process Modelling in Software Quality Management – Towards a Generic Process Metamodel. Software Quality Journal 12, 2004, 265-283.

[Brinkkemper, 96] Brinkkemper, S.: Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology 38(4), 1996, 275-280.

[Brinkkemper, 98] Brinkkemper, S., Saeki, M., Harmsen, F.: Assembly Techniques for Method Engineering. In Advanced Information Systems Engineering. 10th International Conference, CAiSE'98, Pisa, Italy, June 8-12, 1998, Proceedings, B. Pernici, C. Thanos, Eds. Springer Verlag, LNCS 1413, 1998, 381-400.

[Brinkkemper, 01] Brinkkemper, S., Saeki, M., Harmsen, F.: A Method Engineering Language for the Description of Systems Development Methods (Extended Abstract). In Advanced Information Systems Engineering 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001. Proceedings, K.R. Dittrich, A. Geppert, M.C. Norrie, Eds. Springer-Verlag, LNCS 2068, Berlin, 2001, 473-476.

[Brinkkemper, 06] Brinkkemper, S.: Personal Email Communication to Authors, 29 Sept 2006.

[Brooks, 87] Brooks, F.P. Jr.: No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer 20(4), 1987, 10-19.

[Bucher, 07] Bucher, T., Klesse, M., Kurpjuweit, S., Winter, R.: Situational Method Engineering. On the Differentiation of "Context" and "Project Type", in Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 33-48.

[Chou, 02] Chou, S.-C.: A Process Modeling Language Consisting of High Level UML-Based Diagrams and Low Level Process Language. Journal of Object Technology 1(4), 2002, 137-163.

[Chroust, 00] Chroust, G.: Software Process Models: Structure and Challenges. In Software: Theory and Practice – Proceedings, IFIP Congress 2000, Y. Feng, D. Notkin , M.C. Gaudel, Eds., 2000, 279-286.

[Cockburn, 00] Cockburn, A.: Selecting a Project's Methodology. IEEE Software 17(4), 2000, 64-71.

[Constantine, 94] Constantine, L.L., Lockwood, L.A.D.: One Size Does Not Fit All: Fitting Practices to People. American Programmer 7(12), 1994, 30-38

[Cossentino, 05] Cossentino, M., Seidita, V.: Composition of a new process to meet agile needs using method engineering, in SELMAS 2004, R. Choren *et al.* (eds.), LNCS 3390, Springer-Verlag, Berlin, 2005, 36-51

[Cossentino, 06] Cossentino, M., Gaglio, S., Henderson-Sellers, B., Seidita, V.: A Metamodelling-Based Approach for Method Fragment Comparison. In EMMSAD'06 Exploring Modeling Methods for Systems Analysis and Design, J. Krogstie, T. Halpin and E. Proper (eds.), CEUR Workshop Proceedings, Vol 364, 57-70

[Cunin, 01] Cunin, P.-Y., Greenwood, R.M., Francou, L., Robertson, I., Warboys, B.: The Pie Methodology – Concept and Application. In Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001, V. Ambriola, Ed. Springer-Verlag, LNCS 2077, Berlin, 2001, 3–26.

[Deneckère, 98] Deneckère, R., Souveyet, C.: Patterns for Extending An OO Model with Temporal Features. In OOIS'98 Proceedings, C. Rolland, G. Grosz, Eds. Springer-Verlag, London, 1998, 201-218

[Deneckère, 01] Deneckère, R.: Approche D'extension De Methods Fondée Sur L'utilisation De Composants Génériques. PhD Thesis, University of Paris 1-Sorbonne, France, 2001.

[Deneckère, 08] Deneckère, R., Iacovelli, A., Kornyshova, E. and Souveyet, C.: From method fragments to method services, Proceedings of the 13th International Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'08) held in conjunction with the CAiSE'08 Conference, Montpellier, France, June 16-17, 2008, Eds. T. Halpin, E. Proper, J. Krogstie, X. Franch, E. Hunt, R. Coletta, CEUR Workshop Proceedings, Vol. 337, 80-96 (http://www.ceur-ws.org)

[Dorling, 93] Dorling, A.: SPICE: Software Process Improvement and Capability dEtermination, Information and Software Technology, 35(6/7), 1993, 404-406.

[D'Souza, 98] D.Souza, D., Wills, A.C.: Objects, Components, And Frameworks with UML: The Catalysis Approach. Addison-Wesley, Boston, MA, USA, 1998.

[Finkelstein, 94] Finkelstein, A., Kramer, J., Nuseibeh, B.: Software Process Modelling And Technology. Research Studies Press Ltd., John Wiley & Sons Inc., Taunton, England, 1994.

[Fiorini, 01] Fiorini, S.T., Leite, J.C.S.P., De Lucena, C.J.P.: Process Reuse Architecture. In Proceedings of CAiSE2001, K.R. Dittrich, A. Geppert, M.C. Norrie. Eds. Springer-Verlag, LNCS 2068, Berlin, 2001, 284-298.

[Firesmith, 95] Firesmith, D.G., Eykholt, E.M.: Dictionary of Object Technology: The Definitive Desk Reference, SIGS, New York, NY, USA, 603pp, 1995

[Firesmith, 97] Firesmith, D., Henderson-Sellers, B., Graham, I.: OPEN Modeling Language (OML:) Reference Manual. SIGS Books, New York, NY, USA, 276pp, 1997.

[Firesmith, 02] Firesmith, D.G., Henderson-Sellers, B.: The OPEN Process Framework. An Introduction. Addison-Wesley, London, UK, 330pp, 2002.

[Fitzgerald, 03] Fitzgerald, B., Russo, N.L., O'Kane, T.: Software Development Method Tailoring At Motorola, Communications of the ACM 46(4), 2003, 65-70.

[Freeman, 91] Freeman, C., Henderson-Sellers, B.: OLMS – An Object Library Management Support System. In TOOLS 6, J. Potter, M. Tokoro, B. Meyer, Eds. Prentice Hall, Sydney, 1991, 175-180.

[Garcia-Ojeda, 09] Garcia-Ojeda, J.C., DeLoach, S.A., Robby : agentTool Process Editor: Supporting the Design of Tailored Agent-based Processes. Proceedings of the 24th Annual ACM Symposium on Applied Computing to be held at the Hilton Hawaiian Village Beach Resort & Spa Waikiki Beach, Honolulu, Hawaii, USA, March 8 - 12, 2009, ACM Press.

[Gericke 09] Gericke, A., Fill, H.-G., Karagiannis, D., Winter, R.: Situational Method Engineering for Governance, Risk and Compliance Information Systems. In Procs.4th Int. Conf. on Design Science Research in Information Systems and Technology, ACM Press, New York, NY, USA, 2009, Article no. 24.

[Glass, 00] Glass, R.L.: Process Diversity and a Computing Old Wives'/Husbands' Tale. IEEE Software 17(4), 2000, 128-127.

[Glass, 03] Glass, R.L.: Questioning the Software Engineering Unquestionables. IEEE Software 20(3), 2003, 120-119.

[Glass, 04] Glass, R.L.: Matching Methodology to Problem Domain. Communications of the ACM 47(5), 2004, 19-21.

[Gnatz, 01] Gnatz, M., Marschall, F., Popp, G., Rausch, A., Schwerin, W.: Towards a Living Software Process Development Process Based on Process Patterns. In Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001, V. Ambriola. Ed. Springer-Verlag, LNCS 2077, Berlin, 182–202.

[Gonzalez-Perez, 05a] Gonzalez-Perez, C.: Tools for An Extended Object Modelling Environment. In Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems, IEEE Computer Society, Washington DC, 2005, 20-23.

[Gonzalez-Perez, 05b] Gonzalez-Perez, C., Henderson-Sellers, B.: Templates and Resources in Software Development Methodologies. Journal of Object Technology 4(4), 2005, 173-190.

[Gonzalez-Perez, 06a] Gonzalez-Perez, C., Henderson-Sellers, B.: A Powertype-Based Metamodelling Framework. Software and Systems Modeling, 2006, 5(1), 72-90

[Gonzalez-Perez, 06b] Gonzalez-Perez, C., Henderson-Sellers, B.: On the Ease of Extending a Powertype-Based Methodology Metamodel. In Meta-Modelling and Ontologies. Proceedings of the 2nd Workshop on Meta-Modelling, WOMM 2006, LNI Volume P-96, 2006, 11-25.

[Gonzalez-Perez, 07] Gonzalez-Perez, C., Henderson-Sellers, B.: Modelling Software Development Methodologies: A Conceptual Foundation. Journal of Systems Software, 80(11), 2007, 1778-1796

[Gonzalez-Perez, 08a] Gonzalez-Perez, C., Henderson-Sellers, B.: Methodology Enactment Using a Work Product Pool Approach, J. Systems and Software, 81(8), 2008, 1288-1305

[Gonzalez-Perez, 08b] Gonzalez-Perez, C., Henderson-Sellers, B.: Metamodelling for Software Engineering, J. Wiley & Sons, Chichester, 210pp, 2008.

[Graham, 95a] Graham, I.: A Non-Procedural Process Model for Object-Oriented Software Development. Report on Object Analysis and Design, 1(5), 1995, 10-11.

[Graham, 95b] Graham, I.: Migrating to Object Technology. Addison-Wesley, Wokingham, UK, 552pp, 1995.

[Graham, 97] Graham, I., Henderson-Sellers, B., Younessi, H.: The OPEN Process Specification. Addison-Wesley, London, UK, 314pp, 1997.

[Greenwood, 01] Greenwood, R.M., Balasubramaniam, D., Kirby, G., Mayes, K., Morrison, R., Seet, W., Warboys, B., Zirintsis, E.: Reflection and Reification in Process System Evolution: Experience and Opportunity. In Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001, V. Ambriola, Ed. Springer-Verlag, LNCS 2077, Berlin, 2001, 27–38.

[Greiffenberg, 03] Greiffenberg, S.: Methodenentwicklung in Wirtschaft und Verwaltung. Verlag Dr. Kovac, Hamburg, Germany, 2003.

[Haire, 00] Haire, B.: Web OPEN: an extension to the OPEN framework, Capstone project, Faculty of Engineering, University of Technology, Sydney, 2000, 122pp.

[Han, 08] Han, T., Purao, S., Storey, V.C.: Generating large-scale repositories of reusable artifacts for conceptual design of information systems, Decision Support Systems, 45(4), 2008, 665-680

[Harmsen, 94] Harmsen, A.F., Brinkkemper, S., Oei, H.: Situational Method Engineering for Information Systems Projects. In Methods and Associated Tools for the Information Systems Life Cycle. Proceedings of the IFIP WG8.1 Working Conference Cris/94, T.W. Olle, A.A. Verrijn-Stuart, Eds. North Holland, Amsterdam, 1994, 169-194.

[Harmsen, 97] Harmsen, A.F.: Situational Method Engineering. Moret Ernst & Young, Amsterdam, The Netherlands, 1997.

[Hassine, 04] Hassine, L., Ben Ghazala, H.: Une Approche pour la Définition de Méthodes Situationelles d'Assurance de la Qualité Logicielle. Génie Logiciel, 70, 2004, 29-37.

[Henderson-Sellers, 94] Henderson-Sellers, B., Edwards, J.M. BOOKTWO of Object-Oriented Knowledge: The Working Object. Prentice-Hall, Sydney, Australia, 594pp. 1994.

[Henderson-Sellers, 95] Henderson-Sellers, B.: Who Needs An OO Methodology Anyway?, Guest Editorial for Journal of Object-Oriented Programming 8(6), 1995, 6-8.

[Henderson-Sellers, 96a] Henderson-Sellers, B., Bulthuis, A.: The Comma Project. Object Magazine 6(4), 1996, 24-26.

[Henderson-Sellers, 96b] Henderson-Sellers, B., 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, R. Winder: OPEN: Toward Method Convergence? IEEE Computer, 1996, 29(4), 86-89

[Henderson-Sellers, 98a] Henderson-Sellers, B., Bulthuis, A.: Object-Oriented Metamethods. Springer, New York, NY, USA, 158pp, 1998.

[Henderson-Sellers, 98b] Henderson-Sellers, B., Simons, A.J.H., Younessi, H.: The OPEN Toolbox of Techniques. Addison-Wesley, London, UK, 426pp + CD, 1998.

[Henderson-Sellers, 02a] Henderson-Sellers, B., Haire, B., Lowe, D.: Using OPEN's Deontic Matrices for E-Business. In Engineering Information Systems in the Internet Context, C. Rolland, S. Brinkkemper, M. Saeki, Eds., Kluwer Academic Publishers, Boston, USA, 2002, 9-30.

[Henderson-Sellers, 02b] Henderson-Sellers, B., Lowe, D., Haire, B.: OPEN Process Support for Web Development. Annals of Software Engineering 13, 2002, 163-201.

[Henderson-Sellers, 03] Henderson-Sellers, B.: Method Engineering for OO System Development. Communications of the ACM 46(10), 2003, 73-78.

[Henderson-Sellers, 04a] Henderson-Sellers, B., Nguyen, V.P.: Un Outil D'aide a L'ingénierie De Méthodes Reposant Sur L'approche OPEN. Génie Logiciel 70, 2004, 17-28.

[Henderson-Sellers, 04b] Henderson-Sellers, B., Serour, M., Mcbride, T., Gonzalez-Perez, C., Dagher, L.: Process Construction and Customization. Journal of Universal Computer Science, 10(4), 2004, 326-358.

[Henderson-Sellers, 05a] Henderson-Sellers, B.: Creating a Comprehensive Agent-Oriented Methodology – Using Method Engineering and the OPEN Metamodel. In Agent-Oriented Methodologies, B. Henderson-Sellers, P. Giorgini, Eds. Idea Group, Hershey, PA, 2005, 368-397.

[Henderson-Sellers, 05b] Henderson-Sellers, B., Serour, M.K.: Creating a Dual Agility Method – The Value of Method Engineering. Journal of Database Management 16(4), 2005, 1-24.

[Henderson-Sellers, 06a] Henderson-Sellers, B.: Method Engineering: Theory and Practice. In Information Systems Technology and Its Applications. 5th International Conference ISTA 2006. May 30-31, 2006. Klagenfurt, Austria, D. Karagiannis, H.C. Mayr, Eds. Lecture Notes in Informatics (LNI) – Proceedings, Volume P-84, Gesellschaft Für Informatik, Bonn, 2006, 13-23.

[Henderson-Sellers, 06b] Henderson-Sellers, B.: SPI – A Role for Method Engineering. In Euromicro 2006. Proceedings of the 32nd Euromicro Conference on Software Engineering and Advanced Applications (Seaa), IEEE Computer Society Press, Los Alamitos, CA, USA, 2006, 4-5.

[Henderson-Sellers, 07a] Henderson-Sellers, B., Serour, M.K., Gonzalez-Perez, C., Qumer, A.: Improving Agile Software Development by the Application of Method Engineering Practices. In Proceedings IASTED Software Engineering Conference, Innsbruck, 13-15 February 2007.

[Henderson-Sellers, 07b] Henderson-Sellers, B., Gonzalez-Perez, C., Ralyté, J.: Situational Method Engineering: Chunks Or Fragments?, In Proceedings. CAiSE Forum, CAiSE2007, Trondheim, 11-15 June 2007.

[Henderson-Sellers, 08] Henderson-Sellers, B., Gonzalez-Perez, C., Ralyté, J.: Comparison of Method Chunks and Method Fragments for Situational Method Engineering, Proceedings 19th Australian Software Engineering Conference. ASWEC2008, IEEE Computer Society, Los Alamitos, CA, USA, 2008, 479-488

[Henninger, 02] Henninger, S., Ivaturi, A., Nuli, K., Thirunavukkaras, A.: Supporting Adaptable Methodologies to Meet Evolving Project Needs. In Proceedings of the 1st ICSE Workshop on Iterative, Adaptive, And Agile Processes, Orlando, Florida, USA, May 25, 2002.

[Hong, 93] Hong, S., Van Den Goor, Brinkkemper, S.: A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies. In Proceedings of the 26th Hawaii International Conference on System Science, IEEE Computer Society, Los Alamitos, Volume IV, 1993, 689-698.

[Hruby, 00] Hruby, P.: Designing Customizable Methodologies. JOOP Dec 2000, 22-31.

[ISO/IEC, 95] ISO/IEC: Software Life Cycle Processes. ISO/IEC 12207. International Standards Organization/ International Electrotechnical Commission, Geneva, Switzerland, 1995.

[ISO/IEC, 98] ISO/IEC: TRL5504 – Information Technology: Software Process Assessment. Technical Report, International Standards Organization/International Electrotechnical Commission, Geneva, Switzerland, 1998.

[ISO/IEC, 07] ISO/IEC: Software Engineering: Metamodel for Development Methodologies. Iso/Iec 24744. International Standards Organization/International Electrotechnical Commission, Geneva, Switzerland, 2007.

[Jarke, 98] Jarke, M., Pohl, K., Weidenhaupt, K., Lyytinen, K., Marttiin, P., Tolvanen, J.-P., Papazoglou, M.: Meta Modelling: A Formal Basis for Interoperability and Adaptability. In Information Systems Interoperability, B. Krämer, H.-W. Schmidt, Eds. Research Studies Press Ltd./John Wiley & Sons Inc., 1998, 229-263.

[Jarke, 99] Jarke, M., Rolland, C., Sutcliffe, A., Domges, R.: The Nature Requirements Engineering, Shaker-Verlag, Aachen, Germany, 1999.

[Jayaratna, 94] Jayaratna, N.: Understanding and Evaluating Methodologies, Nimsad: A Systemic Approach. Mcgraw-Hill., London, UK, 1994.

[Jeffery, 88] Jeffery, D.R., Basili, V.R.: Validating the Tame Resource Data Model. In Proceedings of the 10th International Conference on Software Engineering, April 11-15 1988 Singapore.

[Jiang, 08] Jiang, L., Eberlein, A., Far, B.H., Mousavi, M.: A methodology for the selection of requirements engineering techniques, Software and Systems Modeling, 7(3), 2008, 303-328

[Karlsson, 01] Karlsson, F., Ågerfalk, P., Hjalmarsson, A.: Method Configuration with Development Tracks and Generic Project Types. In Sixth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, 2001.

[Karlsson, 02] Karlsson, F.: Bridging the Gap – Between Method for Method Configuration and Situational Method Engineering. In Proceedings of Promote IT 2002, 22-24 April 2002, Skövde, Sweden, Knowledge Foundation, Stockholm, 2002.

[Karlsson, 04] Karlsson, F., Ågerfalk, P.J.: Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets. Information and Software Technology 46, 2004, 619-633.

[Karlsson, 05] Karlsson, F., Ågerfalk, P.J.: Method-User-Centred Method Configuration. In Proceedings of the First International Workshop on Situational Requirements Engineering Processes: Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes (SREP'05), Paris France, August 2005, in Conjunction with the Thirteenth IEEE Requirements Engineering Conference (RE'05), J. Ralyté, P.J. Ågerfalk, N. Kraiem, Eds. 31-43.

[Karlsson, 06] Karlsson, F. Wistrand, K.: Combining Method Engineering with Activity Theory: Theoretical Grounding of the Method Component Concept. European Journal of Information Systems, 15, 2006, 82-90

[Karlsson, 07] Karlsson, F., Ågerfalk, P.J.: Multi-grounded Action Research in Method Engineering: The MMC Case, in Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 19-32.

[Karlsson, 09] Karlsson, F., Ågerfalk, P.: Exploring Agile Values in Method Configuration. European Journal of Information Systems, 18, 2009, 300-316.

[Kelly, 96] Kelly, S., Lyytinen, K., Rossi, M.: Metaedit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In Proceedings of the 8th Conf. on Advanced Information Systems Engineering, Y. Vassiliou, J. Mylopoulos, Eds. Springer-Verlag, 1996.

[Kokol, 99] Kokol, P.: Method Engineering and Unified Paediatric Health Care Encounter Design. International Journal of Healthcare Technology and Management 1(3/4), 1999, 401-408.

[Kornyshova, 07] Kornyshova, E., Deneckère, R., Salinesi, C. : Method Chunks Selection by Multicriteria Techniques: An Extension of the Assembly-based Approach. In Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 64-78.

[Kruchten, 99] Kruchten, Ph.: The Rational Unified Process: An Introduction. Addison-Wesley, Reading, MA, USA, 1999.

[Kumar, 92] Kumar, K., Welke, R.J.: Methodology Engineering: A Proposal for Situation-Specific Methodology Construction. In Challenges and Strategies for Research in Systems Development, W.W. Cotterman, J.A. Senn, Eds. John Wiley & Sons: Chichester, UK, 1992, 257-269.

[Lyytinen, 87] Lyytinen, K.: Different Perspectives on Information Systems: Problems and Solutions. ACM Computer Surveys, 19(1), 1987, 5-46.

[Lyytinen, 94] Lyytinen, K., Kerola, P., Kaipala, J., Kelly, S., Lehto, J., Liu, H., Marttiin, P., Oinas-Kukkonen, H., Pirhonen, J., Rossi, M., Smolander, K., Tahvanainen, V.-P., Tolvanen, J.-P.: Metaphor: Metamodeling, Principles, Hypertext, Objects and Repositories. Computer Science and Information Systems Reports, Technical Report TR-7, Department of Computer Science and Information Systems, University of Jyväskylä, Finland, 39pp, 1994.

[MacIsaac, 03] MacIsaac, B.: An Overview of the RUP As a Process Engineering Platform. In Proceedings of the OOPSLA 2003 Workshop on Process Engineering for Object-Oriented and Component-Based Development. Anaheim, CA, 26-30 October 2003, C. Gonzalez-Perez, B. Henderson-Sellers, D. Rawsthorne, Eds. COTAR, Sydney, 43-52

[Madhavji, 91] Madhavji, N.H.: The Process Cycle. Software Engineering Journal 6(5), 1991, 234-242.

[McGregor, 08] McGregor, J.D.: Mix and match, Journal of Object Technology, 7(6), 2008, 7-13

[Mirbel, 02] Mirbel, I., De Rivieres, V.: Adapting Analysis and Design to Software Context: The Jecko Approach. In Proceedings of the 8th International Conference on Object-Oriented Information Systems (OOIS'02), Montpellier, France, September 2-5, 2002, Z. Bellahsène, D. Patel, C. Rolland, Eds. Springer-Verlag, LNCS 2425, 223-228.

[Mirbel, 06a] Mirbel, I.: Method Chunk Federation. In CAiSE'06. 18th Conference on Advanced Information Systems Engineering – Trusted Information Systems. Luxembourg 5-9 June, 2006. Proceedings of the Workshops and Doctoral Consortium, T. Latour, M. Petit, Eds. Namur University Press, Belgium, 407-418.

[Mirbel, 06b] Mirbel, I., Ralyté, J.: Situational Method Engineering: Combining Assembly-Based and Roadmap-Driven Approaches. Requirements Engineering 11, 2006, 58-78.

[Nguyen, 03] Nguyen, V.P., Henderson-Sellers, B.: Towards Automated Support for Method Engineering with the OPEN Approach. In Proceedings of the 7th IASTED Sea Conference, Acta Press, Anaheim, USA, 2003, 691-696.

[Niknafs, 09] Niknafs, A., Asadi, M.: Towards a Process Modeling Language for Method Engineering Support. In Procs. 2009 World Congress on Computer Science and Information Engineering, Eds. M. Burgin, M.H. Chowdhury, C.H. Ham, S.A. Ludwig, W. Su and S. Yenduri. IEEE Computer Society, Los Alamitos, SA, USA, 2009, 674-681

[Nuseibeh, 96] Nuseibeh, B., Finkelstein, A., Kramer, J.: Method Engineering for Multi-Perspective Software Development. Information and Software Technology 38(4), 1996, 267-274.

[Odell, 94] Odell, J.J.: Power Types. Journal of Object-Oriented Programming 7(2)**,** 1994, 8-12.

[Odell, 95] Odell, J.J.: Introduction to Method Engineering. Object Magazine. 1995. 5(5).

[Oivo, 92] Oivo, M., Basili, V.R.: Representing Software Engineering Models: The Tame Goal Oriented Approach. IEEE Transactions on Software Engineering 18(10), 1992, 886-898.

[OMG, 01] OMG: OMG Unified Modelling Language Specification, Version 1.4. OMG Documents Formal/01-09-68 through 80 (13 Documents). 2001, http://www.omg.org, Accessed 12th July 2002.

[OMG, 02] OMG: Software Process Engineering Metamodel Specification, Formal/2002-11-14, Object Management Group, 2002.

[Paulk, 93] Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: The Capability Maturity Model: Version 1.1. IEEE Software 10(4), 1993, 18-27.

[Patel, 04] Patel, C., De Cesare, S., Iacovelli, N., Merico, A.: A Framework for Method Tailoring: A Case Study. In Proceedings of the Second Workshop on Method Engineering for Object-Oriented and Component-Based Development, M. Serour, Ed. Centre for Object Technology Applications and Research, Sydney, 2004, 23-37.

[Perez, 95] Perez, G., El Amam, K., Madhavji, N.H.: Customising Software Process Models. In Proceedings of the 4th EWSPT, Leiden, Holland, March 1995, 70-78.

[Plihon, 98] Plihon, V., Ralyté, J., Benjamen, A., Maiden, N.A.M., Sutcliffe, A., Dubois, E., Heymans, P.: A Reuse-Oriented Approach for the Construction of Scenario Based Methods. In 5th Int. Conf. on Software Process (ICSP'98), Chicago, Illinois, USA, 1998.

[Punter, 96] Punter, H.T., Lemmen, K.: The MEMA Model: Towards a New Approach for Method Engineering. Information and Software Technology 38(4), 1996**,** 295-305.

[Qumer, 07] Qumer, A., Henderson-Sellers, B.: Construction of an agile software product enhancement process by using an agile software solution framework (ASSF) and situational method engineering, Procs. 31st Annual International Computer Software and Applications Conference, Beijing, China, 23-27 July 2007 Volume 1, IEEE Computer Society, 2007, 539-542

[Ralyté, 99] Ralyté, J.: Reusing Scenario Based Approaches in Requirements Engineering Methods: Crews Method Base. In Proceedings of the 10th Int. Workshop on Database and Expert Systems Applications (DEXA'99), 1st Int. Rep'99 Workshop, Florence, Italy, 1999.

[Ralyté, 01a] Ralyté, J., Rolland, C.: An Assembly Process Model for Method Engineering. In Advanced Information Systems Engineering, K.R. Dittrich, A. Geppert, M.C. Norrie, Eds. Springer, LNCS2068, Berlin, 2001, 267-283.

[Ralyté, 01b] Ralyté, J., Rolland, C.: An Approach for Method Engineering. In Proceedings of the 20th International Conference on Conceptual Modelling (ER2001), Springer-Verlag, LNCS 2224, Berlin, 2001, 471-484.

[Ralyté, 01c] Ralyte, J.: Ingénierie Des Methods Par Assemblage De Composants. Thèse De Doctorat En Informatique De L'université Paris 1, Janvier 2001, France.

[Ralyté, 02] Ralyté, J.: Requirements Definition for the Situational Method Engineering. In Engineering Information Systems in the Internet Context, C. Rolland, S. Brinkkemper, M. Saeki, Eds. Kluwer Academic Publishers, Boston, USA, 2002, 127-152.

[Ralyté, 03] Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Method for Situational Method Engineering. In Advanced Information Systems Engineering. 15th International Conference, CAiSE 2003, Klagenfurt, Austria, June 16-18, 2003, Proceedings, J. Eder, M. Missikoff, Eds. Springer-Verlag, LNCS 2681, 2003, 95-110.

[Ralyté, 04a] Ralyté, J.: Towards Situational Methods for Information Systems Development: Engineering Reusable Method Chunks. In Proceedings of the International Conference on Information Systems Development (ISD'04), Vilnius Technika, 2004, 271-282.

[Ralyté, 04b] Ralyté, J., Rolland, C., Deneckère, R.: Towards a Meta-Tool for Change-Centric Method Engineering: A Typology of Generic Operators. In Proceedings of CAiSE 2004, A. Persson, J. Stirna, Eds. Springer-Verlag, LNCS 3084, Berlin, 2004, 202-218.

[Ralyté, 05] Ralyté, J., Rolland, C., Ben Ayed, M.: An Approach for Evolution-Driven Method Engineering'. In Information Modeling Methods and Methodologies' (Advanced Topics in Database Research), J. Krogstie, T. Halpin, K. Siau, Eds. Idea Group, Hershey, PA, 2005, 80-100.

[Ralyté, 07] Ralyté, J.; Brinkkemper, S., Henderson-Sellers, B. (eds.), Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland, IFIP Series, Vol. 244, Springer, Berlin, 380pp, 2007.

[Rausch, 05] Rausch, A., Höhn, R., Höppner, S.: Das V-Modell Xt, Springer, Berlin, Germany, 2005.

[Rolland, 95] Rolland, C., Souveyet, C., Moreno, M.: An Approach for Defining Ways-of-Working. Information Systems 20(4), 1995, 295-305.

[Rolland, 96] Rolland, C., Prakash, N.: A Proposal for Context-Specific Method Engineering. In Method Engineering. Principles of Method Construction and Tool Support. Proceedings of IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA, S. Brinkkemper, K. Lyytinen, R.J. Welke, Eds. Chapman & Hall, London, 191-208.

[Rolland, 98a] Rolland, C.: A Comprehensive View of Process Engineering. In Advanced Information Systems Engineering: 10th International Conference, CAiSE'98, Pisa, Italy, June 1998. Proceedings, B. Pernici, C. Thanos, Eds. Springer-Verlag, LNCS 1413, Berlin, 1998, 1-24.

[Rolland, 98b] Rolland, C., Plihon, V., Ralyté, J.: Specifying the Reuse Context of Scenario Method Chunks. In Advanced Information Systems Engineering 10th International Conference, CAiSE'98, Pisa, Italy, June 8-12, 1998, Proceedings, B. Pernici, C. Thanos, Eds. Springer-Verlag, LNCS1413, Berlin, 191-218.

[Rolland, 99] Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. Requirements Engineering Journal 4(4), 1999, 169-187.

[Rolland, 02] Rolland, C.: A User Centric View of Lyee Requirements. In New Trends in Software Methodologies, Tools and Techniques, H. Fujita, P. Johannesson, Eds. IOS Press, Tokyo, 2002.

[Rolland, 09] Rolland, C.: Method Engineering: Towards Methods as Services. Software Process Improvement and Practice, 14, 2009, 143-164.

[Rossi, 95] Rossi, M.: The Metaedit CAME Environment. In Proceedings of Metacase 95, University of Sunderland Press, Sunderland, UK, 1995.

[Rossi, 98] Rossi, M.: Advanced Computer Support for Method Engineering – Implementation of CAME Environment in Metaedit+. Dissertation, Jyväskylä Studies in Computer Science, Economics and Statistics, Vol. 42, University of Jyväskylä, Finland, 1998.

[Rossi, 04] Rossi, M., Ramesh, B., Lyytinen, K., Tolvanen, J.-P. 2004. Managing Evolutionary Method Engineering by Method Rationale. Journal of the Association of Information Systems 5(9), 356-391.

[Rumbaugh, 91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, USA, 1991.

[Rupprecht, 00] Rupprecht, C., Funffinger, M., Knublauch, H., Rose, T.: Capture and Dissemination of Experience About the Construction of Engineering Processes. In: Proceedings of the 12th Conference on Advanced Information Systems Engineering (CAiSE), Springer-Verlag, LNCS 1789, Berlin, 2000, 294-308.

[Saeki, 93] Saeki, M., Iguchi, K., Wen-yin, K. and Shinohara, M.: A Meta-model for Representing Software Specification & Design Methods. In Procs. IFIP WG8.1 Conf on Information Systems Development Process, Come, 1993, 149-166

[Saeki, 03] Saeki M.: Embedding Metrics into Information Systems Development Methods: An Application of Method Engineering Technique. In CAiSE'03, Proceedings, Springer, LNCS 2681, Berlin, 2003, 374-389.

[Scott, 01] Scott, L., Carvalho, L., Jeffery, R., D'ambra, J.: An Evaluation of the Spearmint Approach to Software Process Modelling. In Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001, V. Ambriola, Ed. Springer-Verlag, LNCS 2077, Berlin, 2001, 77–89.

[Seidewitz, 03] Seidewitz, E.: What models mean. IEEE Software, 20, 2003, 26–31.

[Seidita, 07] Seidita, V., Ralyté, J., Henderson-Sellers, B., Cossentino, M., Arni-Bloch, N.: A Comparison of Deontic Matrices, Maps and Activity Diagrams for the Construction of Situational Methods. In Proceedings of the CAiSE Forum, CAiSE2007, Trondheim, June 2007.

[SEMAT, 10] Software Engineering Method and Theory, http://www.semat.org

[Serour, 02] Serour, M., Henderson-Sellers, B., Hughes, J., Winder, D., Chow, L.: Organizational Transition to Object Technology: Theory and Practice. In Object-Oriented Information Systems, Z. Bellahsène, D. Patel, C. Rolland, Eds. Springer-Verlag, LNCS 2425, Berlin, 2002, 229-241.

[Serour, 04a] Serour, M.K., Henderson-Sellers, B.: Introducing Agility: A Case Study of Situational Method Engineering Using the OPEN Process Framework. In Proceedings of the 28th Annual International Computer Software and Applications Conference. COMPSAC 2004, IEEE Computer Society Press, Los Alamitos, CA, USA, 2004, 50-57.

[Serour, 04b] Serour, M.K., Dagher, L., Prior, J., Henderson-Sellers, B.: OPEN for Agility: An Action Research Study of Introducing Method Engineering Into a Government Sector. In Proceedings of the 13th International Conference on Information Systems Development. Advances in Theory, Practice and Education, O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic, S. Wrycza, Eds. Vilnius Gediminas Technical University, Vilnius, Lithuania, 2004, 105-116.

[Standards Australia, 04] Standards Australia.:. Standard Metamodel for Software Development Methodologies. As 4651-2004, Standards Australia, Sydney, 2004.

[Stapleton, 97] Stapleton, J.: DSDM: The Method in Practice. Addison-Wesley, Reading, MA, USA, 1997.

[Stőrrle, 01] Stőrrle, H.: Describing Process Patterns with UML (Position Paper). In Software Process Technology, Proceedings of the 8th European Workshop, EWSPT 2001, V. Ambriola, Ed. Springer-Verlag, LNCS 2077, Berlin, 2001, 173–181.

[Tasharofi, 07] Tasharofi, S., Ramsin, R.: Process patterns for agile methodologies, in Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 222-237

[ter Hofstede, 97] ter Hofstede, A.H.M., Verhoef. T.F.: On the Feasibility of Situational Method Engineering. Information Systems 22(6/7), 1997, 401-422.

[Tolvanen, 93] Tolvanen, J.-P., Marttiin, P., Smolander, K.: An Integrated Model for Information Systems Modelling. In Proceedings of the 26th Annual Hawaii International Conference on Systems Science, J.F. Nunamaker, R.H. Sprague, Eds. IEEE Computer Society Press, Los Alamitos, CA, USA, 1993.

[Tolvanen, 96] Tolvanen, J.-P., Rossi, M., Liu, H.: Method Engineering: Current Research Directions and Implications for Future Research. In Method Engineering. Principles of Method Construction and Too Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA, S. Brinkkemper, K. Lyytinen, R.J. Welke, Eds. Chapman & Hall, London, 1996, 296-317.

[Tolvanen, 98] Tolvanen, J.-P.: Incremental Method Engineering with Modeling Tools. Dissertation, Jyväskylä Studies in Computer Science, Economics and Statistics, Vol. 47, University of Jyväskylä, Finland, 301pp, 1998.

[Tran, 05] Tran, H.N., Boulette, B., Dong, B.T.: A Classification of Process Patterns. In Proceedings of the International Conference on Software Development, Reykjavik, Iceland, May 27 – June 1 2005.

[van der Hoef, 95] van der Hoef, R. et al. Situatie, Scenario En Succes. Memoranda Informatica, International Research Report, University of Twente, The Netherlands, 1995.

[van de Weerd, 05] van de Weerd, I., Souer, J., Versendaal, J., Brinkkemper, S.: Situational Requirements Engineering of Web Content Management Implementation. In Proceedings of the First International Workshop on Situational Requirements Engineering Processes: Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes (SREP'05), Paris France, August 2005, in Conjunction with the Thirteenth IEEE Requirements Engineering Conference (RE'05), J. Ralyté, P.J. Ågerfalk, N. Kraiem, Eds. 13-30.

[van de Weerd, 06] van de Weerd I., Brinkkemper, S., Souer, J., Versendaal, J.: A Situational Implementation Method for Web-Based Content Management System-Applications: Method Engineering and Validation in Practice. Software Process: Improvement and Practice, 11(5), 2006, 521-538.

[van de Weerd, 07] van de Weerd, I., ven de Weerd, S., Brinkkemper, S., Developing a reference model for game production by method comparison, in Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland (eds. J. Ralyté, S. Brinkkemper and B. Henderson-Sellers), IFIP Series, Vol. 244, Springer, Berlin, 2007, 313-327

[van Slooten, 93] van Slooten, K., Brinkkemper, S.: A Method Engineering Approach to Information Systems Development. In Information Systems Development Process Proceedings IFIP WG8.1, N. Prakash, C. Rolland, B. Pernici, Eds. Elsevier Science Publishers B.V., North-Holland, 1993.

[van Slooten, 96] van Slooten, K., Hodes, B.: Characterizing Is Development Projects, In Proceedings of IFIP TC8 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support, S. Brinkkemper, K. Lyytinen, R. Welke, Eds. Chapman&Hall, Great Britain, 1996, 29-44.

[Vessey, 94] Vessey, I., Glass, R.L.: Application-Based Methodologies: Development by Application Domain. Information Systems Management Fall 1994.

[Waller, 06] Waller, V. Johnston, R.B., Milton, S.K.: Development of a Situation Information Systems Analysis and Design Methodology: A Health Care Setting. In Proceedings of the European and Mediterranean Conference on Information Systems 2006 (EMCIS2006), Z. Irani, O.D. Sarikas, J. Llopis, R. Gonzalez, J. Gasco, Eds. Cd, Brunel University, West London, Paper C94, 8pp, 2006.

[Wistrand, 04] Wistrand, K., Karlsson, F.: Method Components – Rationale Revealed. In Advanced Information Systems Engineering 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings, A. Persson, J. Stirna, Eds. Springer-Verlag, LNCS 3084, Berlin, 2004, 189-201.

[Xu, 02] Xu, P., Ramesh, B.: A Tool for the Capture and Use of Process Knowledge in Process Tailoring. In Proceedings of the 26th Hawaii International Conference on Systems Sciences (HICSS'03), IEEE Computer Society Press, Los Alamitos, CA, USA, 2002.

[Yourdon, 99] Yourdon, E.: Software Process for the 21st Century. Cutter IT Journal, 12(9), 1999, 12-15.

[Zowghi, 05] Zowghi, D., Firesmith, D.G., Henderson-Sellers, B.: Using the OPEN Process Framework to Produce a Situation-Specific Requirements Engineering Method. In Proceedings of the First International Workshop on Situational Requirements Engineering Processes: Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes (SREP'05), Paris France, August 2005, in conjunction with the Thirteenth IEEE Requirements Engineering Conference (RE'05), J. Ralyté, P.J. Ågerfalk, N. Kraiem, Eds. 2005, 59-74.