Harmsen, F., S. Brinkkemper, H. Oei, Situational Method Engineering for Information System Project Approaches. In: A.A. Verrijn Stuart and T.W. Olle (Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Proceedings of the IFIP WG 8.1 Working Conference, Maastricht, Netherlands, September 1994, IFIP Transactions A-55, North-Holland, 1994, ISBN 0-444-82074-4, pp. 169-194.

Situational Method Engineering for Information System Project Approaches

Frank Harmsen, Sjaak Brinkkemper, Han Oei

Centre for Telematics and Information Technology, University of Twente,
P.O. Box 217, 7500 AE Enschede, the Netherlands,
E-mail {harmsen, sjbr, oei}@cs.utwente.nl

ABSTRACT

The purpose of this paper is to give an overview of Situational Method Engineering, which is a type of method engineering. Situational Method Engineering aims at harmonisation of methods by providing rules to configure project-specific methods out of fragments from existing standard methods. It stems from the need for controlled flexibility, which is a compromise between the ad-hoc tuning of methods and the application of rigid, standard methods. Method fragments are stored in a method base, which constitutes the central repository of a Computer Aided Method Engineering (CAME) tool. The CAME tool is used to manipulate method fragments in order to configure project-specific methods. We present a model of the method base, as well as operations on method fragments and examples of required consistency rules for a configured method. Moreover, we discuss the spectrum of approaches to Situational Method Engineering. The paper is concluded with a description of the CAME tool under development.

Keyword Codes: D.2.0; H.1.0; K.6.3

Keywords: Software Engineering, General; Information Systems, Models and Principles, General; Computing Milieux, Management of Computing and Information Systems, Software Management

TABLE OF CONTENTS

1. INTRODUCTION

Methods for information systems development are always of a generic nature. Guidelines, techniques and steps in the method handbooks depart from an average project in an idealised, fixed context to which the method is being applied. Reality is different. Methods are never followed literally, they are tuned to the situation at hand. Steps are left out, or included, milestones and deliverables are settled. Knowledge and experience of the project team determine the structure of the development process and the resulting products in order to deliver the desired information system (IS). All kinds of project factors related to the technology, the development expertise, the external factors and application domain characteristics induce an approach specific for the project.

From a methodological standpoint (taking methodology as the study of methods) is the current situation in the area of methods for information systems development (ISD) frankly disastrous. The recent interest into the paradigm of object-orientation has yielded numerous new different methods (among others [Coad 90], [Rumbaugh 91] and [Wirfs-Brock 90]) requiring comparison studies, such as [Hong 93], to reveal the strengths and weaknesses of the methods to the poor practitioner.

There will always be new emergent paradigms and technology break-throughs: client-server architectures, groupware/CSCW, workflow management, and multi-media are some of the recent ones. Will these also require new methods? The study of Hong et al. [Hong 93] shows that large parts of the new methods are taken from other methods and techniques. This gives rise to a new thinking in the IS methodology: method engineering. Configuration of ISD methods from proven parts of existing methods is the kernel of method engineering. This paper will focus on a particular subarea, namely the construction of project specific methods, so-called situational method engineering. The importance of the notion of scenario, which is a project-specific method, was also recognised in chapter two of [Olle 91]. The scenario philosophy has been further elaborated by Kumar and Welke, who introduced Methodology Engineering, being a method to develop and implement methods [Kumar 92]. A method representation model to analyse and compare existing methods, as well as a tool to manipulate that model in order to build methods, was presented in [Heym 93]. In [Saeki 93], a data base called Method Base is described from which several complete methods can be selected. The Viewpoints approach considers also the integration of several methods and tool components into one method [Nuseibeh 92]. We will first discuss this new advancement against the background of harmonisation and standardisation of ISD methods.

1.1.Harmonisation and Standardisation of methods

To date numerous methods, techniques and tools coexist in the information system society, all aiming at the structuring and support of the information system adaptation or development process. Though there may be –good and bad– reasons for this proliferation of methods and tools, the situation is that it is hard for an organisation to compare a set of methods, techniques and tools on a firm basis, and to select or construct a method that is appropriate or suitable for a specific information system project, from this "Methodology Jungle", a term introduced in [Avison 88].

To resolve this problem, various frameworks for comparison and/or selection have been developed. These frameworks differ in their purpose and underlying philosophy, scope and perspective. Some frameworks aim at standardisation, others provide tools and techniques for harmonisation rather than standardisation. In case of standardisation, one or a few methods are selected and proposed as a standard for any information system project to overcome the proliferation of methods. In case of harmonisation, one does not select a fixed standard, but aims at an open transformation scheme between methods. This transformation scheme may serve for comparison, integration, or evolution of methods [Oei 94].

The scope of a framework can be methods covering the complete life-cycle, or be limited to part of these methods, i.e. techniques, languages, and tools, which are often restricted to particular phases of the life-cycle only (e.g. planning, analysis, design or maintenance). Furthermore, different perspectives can be distinguished; a framework developed from a user's perspective differs from a framework developed from a modeller's perspective. Different scopes and perspectives also require and allow different levels of formality. For example, a technique for database specification for a database specialist requires and allows much more formality than a requirements specification technique for an information analyst or a project planning technique for a project manager.

The reference framework of the CRIS task group of IFIP WG 8.1 ([Olle 91]) provides a unified view on various Information Systems Development Methods (ISDMs) along the dimension of development phase (planning, analysis, and design) versus modelling perspective (data, process, behaviour). The reference framework, being a contribution to the methodology of information systems, can be used for the assessment of ISDMs.

An approach aiming at a more formal comparison of techniques is the Meta Model Hierarchy-framework [Oei 92; Oei 94]. This framework provides a technique for comparison, integration, and evolution of modelling languages on the basis of a basic set of transformations being defined among meta models of these languages. This basic set of transformations is based upon application independent criteria: for instance, the genericity, liberality and expressiveness of languages. The Meta Model Hierarchy-framework aims at harmonisation rather than standardisation, because it does not select one or a few techniques as a standard, but provides an open transformation scheme between techniques.

The FRISCO task group of IFIP WG 8.1 [Lindgreen 90] aims at a framework for fundamental information system concepts. The philosophy of FRISCO is that it is essential to understand the fundamental concepts behind information systems, in order to improve the understanding, use and development of information systems as part of an organisation.

Euromethod [Euromethod 93] is developed from a practitioner's point of view. It aims at the improvement of the customer-supplier relationship during an information system development project. For this purpose, Euromethod serves as a communication-bridge between the terminology, concepts and methods on the customer side and those on the supplier side. Therefore it aims at standardisation on the communication level without forcing the parties to leave their own methods on the development level.

All these frameworks aim at either standardisation or harmonisation of methods or parts of methods. The major benefit of standardisation is that it improves communicability among all information system parties. However, if standardisation leads to rigidity, flexibility is in danger to tune methods or method products to its project-specific needs. Also the gain of standardisation is completely dependent on the acceptance of the standard. History has learned that never a 100% acceptance will be reached, and that the acceptance level will fluctuate.

The Situational Method Engineering approach we will discuss in this paper, tries to take the best of two worlds. We aim at the harmonisation of methods by providing a discipline to construct methods out of existing, proven method fragments. These methods are tuned to the situation in which they are applied. That is, we try to both preserve the benefits of standardisation by definition and reuse of so-called method fragments, and to improve flexibility by providing a means –the Situational Method Engineering discipline– to tune a method to the project-specific needs. The scope of Situational Method Engineering is in principle the complete life-cycle. Furthermore, both the development perspective and the project management perspective are considered. Finally, the level of formality will be determined by what is required and allowed for each aspect of the Situational Method Engineering discipline.

1.2. Scope of Method Engineering

We consider an *Information Systems Development Method* (ISDM) as a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their effort to implement a new information system [Avison 88]. Note that Euromethod introduces the more appropriate term

Information Systems Adaptation, meaning the modification, enhancement, improvement, and/or automation of an information system to fulfil the ever changing needs of an organisation [Euromethod 93]. However, the word *development* in ISDM is kept for recognizability. An ISDM should be *integrated*, i.e. should comprise not only the *steps* to be carried out, but also the supporting *tools*, and descriptions of the *products* to be delivered. However, since very few ISDMs and CASE tools are completely integrated, we continue to speak about ISDMs *and* CASE tools. Moreover, the terms ISDM and *method* are used interchangeably. ISDMs tuned to a specific situation are called *situational methods*.

We consider an *approach* to be a way to perform an *IS project*. An IS project is a temporary organisation, with an adapted information system as goal. This adaptation stems from a certain *IT-need*, formulated by an organisation.

The term *methods engineering* was first introduced in mechanical engineering as the co-ordinated and systematic approach of improving work methods [Maynard 39]. We define *method engineering* [Kumar 92; Heym 93; Slooten 93] as the engineering discipline to design, construct and adapt Information Systems Development Methods, including CASE tools. *Situational Method Engineering* is the area of method engineering focusing on situational methods.

From the examples of method engineering studies we have reviewed, we conclude that method engineering can be viewed on three levels: the macro level of systems development *organisations*, the meso level of systems development *projects*, and the micro level of systems development *techniques*. We will here only deal with the systems development projects, but in related studies we deal with the other types of method engineering.

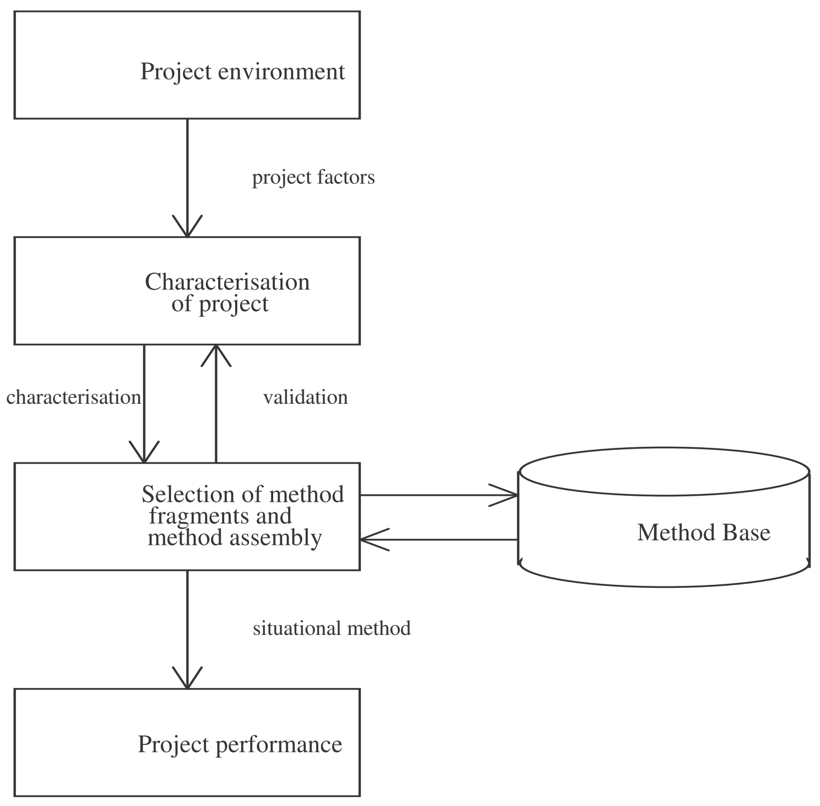


Figure 1.1 The process of configuration of situational methods

The process of configuring a situational method is depicted in figure 1.1, which is an adaptation of a similar figure in [van Slooten 93]. Starting point is a given, dynamic and evolving project environment, being part of a larger organisational setting. The project environment includes the existing information infrastructure, the users, the organisational culture of both the supplier organisation and the customer organisation. Project or contingency factors, such as application characteristics, external factors, technical factors, and the available development expertise, are in some way or another determined forming a project characterisation. This characterisation is input to the process of selecting suitable method fragments from the method base and assembling them into a situational method. As this process involves human operations the resulting method requires validation with respect to the project environment. The method is applied in the project, where it may turn out to be necessary to refine or adapt the method. This may be due to the changing project environment or to the clarification of the project characterisation, i.e. project contingencies that were not clear before can now be rated. We did not include in figure 1.1 the experience accumulation of performed methods into the method base. Parts of methods that are evaluated by the project team as very powerful can be formalised and entered into the method base by the methods administrator.

Figure 1.2 depicts the relationship between project, approach, ISDM, and situational method on the basis of the evolution of ISDMs and situational methods. We see that standard ISDMs (M1, M7) arise from - more or less *ad-hoc* - practical approaches (A2, A3, A6, A7, A8) followed to perform information systems development projects. This is how in the seventies methods like Structured Analysis [Yourdon 89] and JSD [Jackson 83] arose. However, these standard methods are adapted, in order to meet the requirements of the situation at hand or the current state of technology. The result of such an adaptation is a situational method (S1, S2). When a situational method has been used in a lot of projects, it is possible that it becomes a new standard ISDM (M3). For instance, most object-oriented methods evolved from situational methods. Often, these situational methods were adaptations of available standard methods, because object-oriented programming languages were employed. In this way, standard methods emerged like Object Oriented Analysis/Object Oriented Design [Coad 91], being a further development of Modern Structured Analysis supplemented with object-oriented concepts.

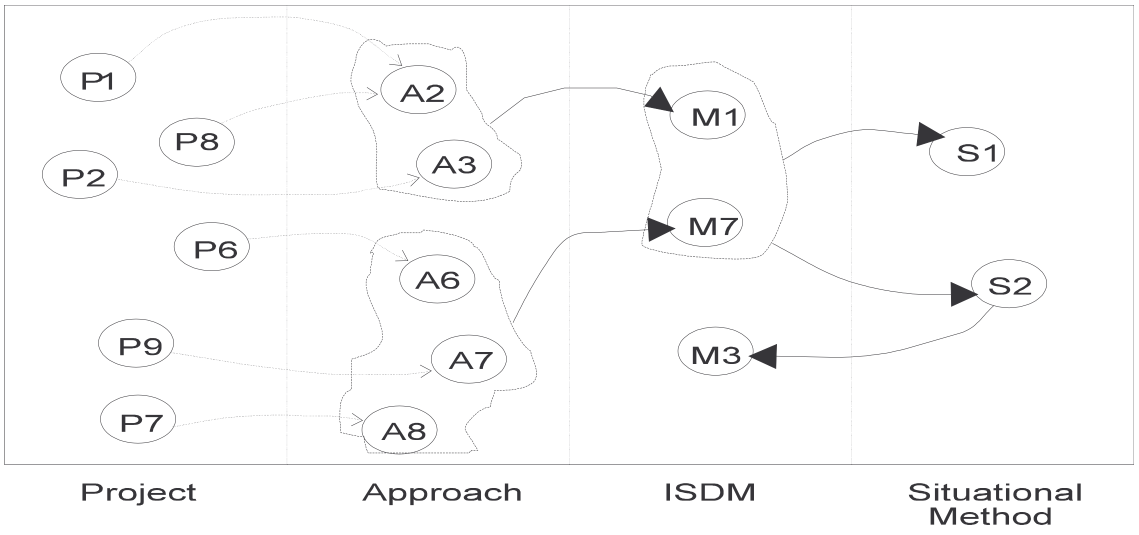


Figure 1.2 The ISDM life cycle

Note that arrows with small arrowheads depict "is conducted with" relationships, whereas the other arrows depict "evolves" relationships.

1.3.The Principle of Controlled Flexibility

Situational Method Engineering stems from the observation that standard information system development methods cannot anticipate adequately to new developments and to specific situations, which makes the current practice of the tuning of existing methods almost inevitable. However, this ad-hoc tuning affects the advantages of using a method, such as standardisation, control, effectiveness and efficiency, in a negative way. Therefore, it is necessary to strike the golden mean between those known benefits, which we consider as the prerequisites of Situational Method Engineering, and flexibility. We believe that this can be achieved by constructing so-called *situational* methods, with standardised and proven building blocks, in a controlled and formalised way, and making use of a uniform terminology. This is how we want to obtain *controlled flexibility*.

Situational Method Engineering contributes to the realisation of the following ISDM requirements:

- **Flexibility.** Situational Method Engineering pre-assumes that the method to be used in a certain information systems development project is *situational*, that is, completely tuned to the project situation at hand, resulting in an approach that is as flexible as possible.
- **Experience accumulation**. The full but controlled adaptability of the method allows for the addition of project experience. This can be achieved by adapting the method building blocks, which are stored in a central data base. This experience can then be used again in new projects.
- **Integration and communication**. In Situational Method Engineering, no difference is made between method and supporting tools; it is a necessary prerequisite that they should be *integrated*. All tools are based on one common Systems Engineering repository, in which also the building blocks of methods are stored. Besides the tools that are usually applied in projects, such as CASE tools, word processors, and project management tools, we envision CAME tools [Kumar 92; Harmsen 93; Heym 93], supporting the construction of the situational method.
- **Quality**. The fact that flexibility should be *controlled* guarantees that the constructed situational method meets the same quality requirements as standard methods do. The flexibility and open architecture of the method allows also for an easy integration of simulation tools and formal specification languages. The standardisation of the method building blocks facilitates the measurement of important project characteristics.

Figure 1.3 depicts the philosophy of controlled flexibility. Harmonisation is obtained by harmonising the building blocks of methods, thus getting the best of both worlds: on the one hand the flexible ways to develop information systems, and on the other hand the rigid, standard approaches. Similar outcome was obtained by the comparative study into the rigidity of rule checking mechanisms in CASE tools [Vessey 92]. The study positioned CASE tools in a spectrum consisting of rigid, flexible and guiding. Preference was shown for the middle of the spectrum.

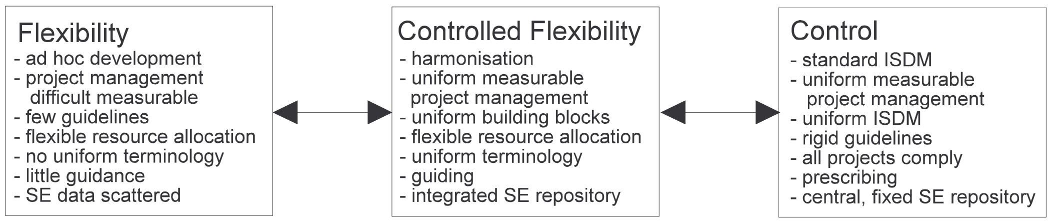


Figure 1.3 Controlled flexibility: harmonisation between flexibility and standardisation

1.4.Overview of the paper and case description

This paper is organised as follows. Section 2 introduces the notion of method fragment. Relationships between method fragments, operations with method fragments, and consistency rules are outlined. Moreover, a summary of the formalisation of method fragments is described. In section 3 overall approaches to Situational Method Engineering are described. Situational Method Engineering is placed in a historical context, and differences with more traditional approaches to information systems development are outlined. Section 4 deals with computerised support for Situational Method Engineering. The paper concludes with a number of conclusions and a description of further research.

Throughout this paper, the theory is illustrated by examples that are taken from a modified Inventory Control and Purchase case [Olle 88]. The description of this case is the following.

The XYZ company has an inventory control system, running on a mainframe computer. However, maintenance costs of the hardware and software are growing fast, and users are complaining about the rudimentary user interface. So, the management of XYZ decides to hire ABC management consultants, and instructs them to start a new ICPS (Inventory Control and Purchase System) project. In this project, the current system should be adapted, with at least the following requirements:

- *The system should have an average response time below 1 second;*
- *The system should run under OS/2, version 2 or higher;*
- *The system should have a functionality that is at least the same as the current system;*
- *The total costs of the project should not exceed 100,000 ECU;*
- *The data bases of the current inventory control system should be transferred without loss of any data.*

2. FUNDAMENTALS OF SITUATIONAL METHOD ENGINEERING

The building blocks of a situational method are called *method fragments*. In principle, any coherent product, activity, or tool being part of an existing generic or situational method is a method fragment. This concept will be described more comprehensively in sub-section 2.1. A special kind of method fragment is a *method outline*. This is a method for which only the main steps, products, or tools have been defined, leaving room for a further project-specific elaboration. An example of a method outline for software package selection, which is something that will be performed in the ICPS project, is shown in figure 2.1.

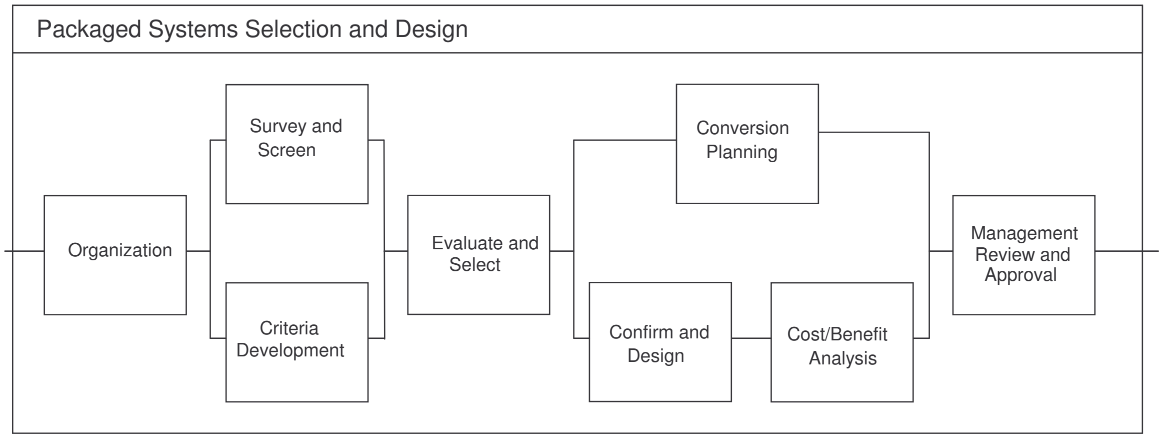


Figure 2.1 Method outline for packaged systems selection and design (taken from [Method/1 92])

Method fragments for the construction of a situational method have to be defined and inserted into a *method base*. A method base is the central repository of a *Computer Aided Method Engineering* (*CAME*) tool. Such a tool provides support for selection and manipulation of method fragments. Ideally, a CAME tool supports the entire process of Situational Method Engineering [Slooten 93]. CAME tools will be discussed in section five. For Situational Method Engineering two new functions are required: the *method engineer*, who is responsible for the configuration of the situational method, and the *methods administrator*, who is responsible for the contents of the method base. Both roles are to be supported by the envisioned CAME tool.

2.1. Types of method fragments

In order to be able to describe both process and data perspective of methods, we distinguish between *process fragments* and *product fragments*. Product fragments are the products and sub-products to be delivered by a method, such as deliverables, milestone documents, models, diagrams, etc. Process fragments represent the stages, activities and tasks to be carried out in order to produce product fragments. Descriptions of product fragments are often referred to as *meta data models,* descriptions of process fragments are also called *meta process models* [Brinkkemper 90; Marttiin 93]. Examples are shown in figure 2.2 and 2.3.

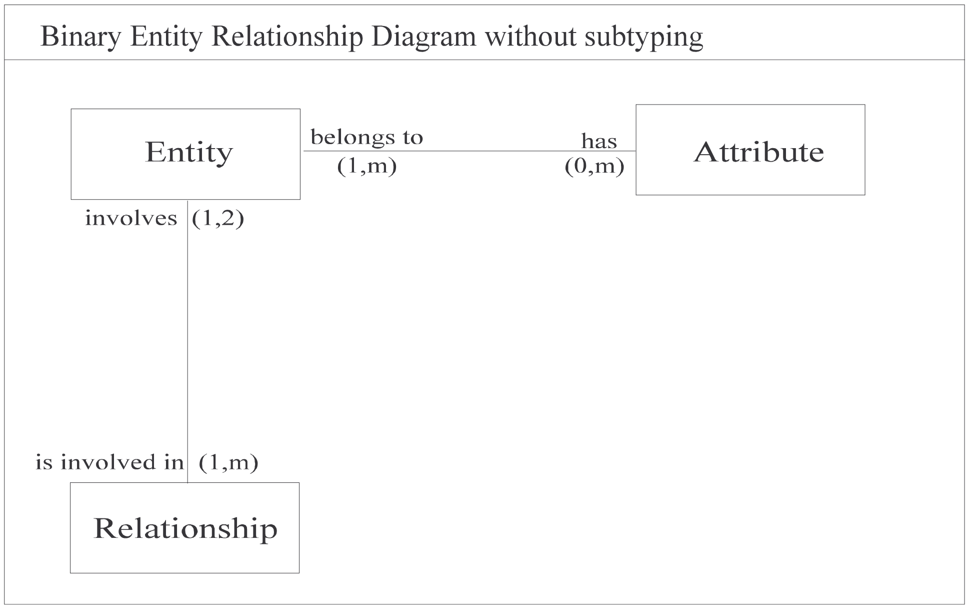


Figure 2.2 Description of a product fragment

Figure 2.2 is a description of the product fragment “Binary Entity Relationship Diagram without subtyping”, a simple variant of ERD. Note that, for the notation of meta data models, we use the Extended Entity Relationship model as described in [Batini 92]. We see that this product fragment contains three concepts, The concept *Entity* is involved in one or more *relationships*, and has zero or more *attributes*.

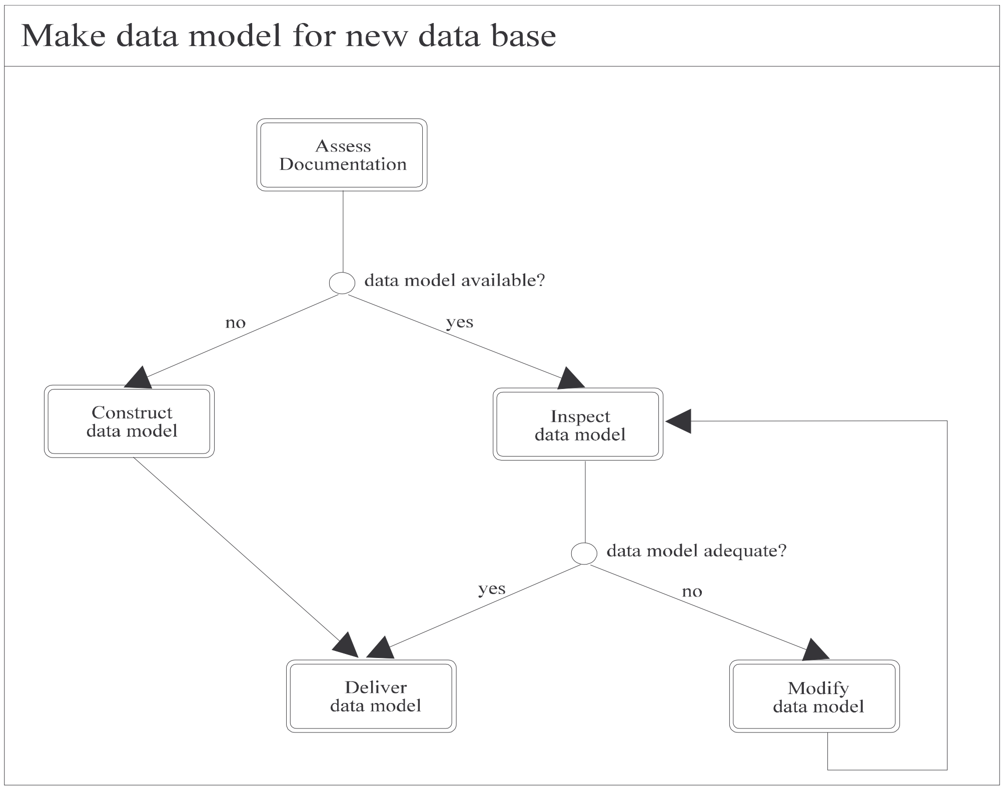


Figure 2.3 Description of a process fragment

Figure 2.3 is a Task Structure Diagram [Bots 87] of the process fragment "Convert Data Model". Rounded rectangles denote tasks, circles denote decisions, arrows denote triggers. This activity starts with the assessment of available system documentation. If the data model is available, then it is inspected, otherwise a new data model is constructed. The other tasks are self-evident. All tasks of this activity are decomposable

Fragments can be considered on a conceptual level, but can also be considered as descriptions of tools or part thereof. The first type is called *conceptual* method fragment, the latter type *technical* method fragment. This is similar to the distinction between the conceptual level and the internal level in the ANSI/SPARC information systems architecture. Conceptual method fragments represent methods or part thereof, such as model descriptions and activities. Technical method fragments are required in order to be able to include components of CASE tools, such as diagram editors and repositories, in the situational method. Besides these two types, we also distinguish *external* method fragments, for views of project members on the method. This will be discussed in a forthcoming paper.

The various types of method fragments and their relationships are depicted in figure 2.4. Note that this diagram represents the data model of a method base (a meta-meta model), albeit in a simplified manner for ease of presentation.

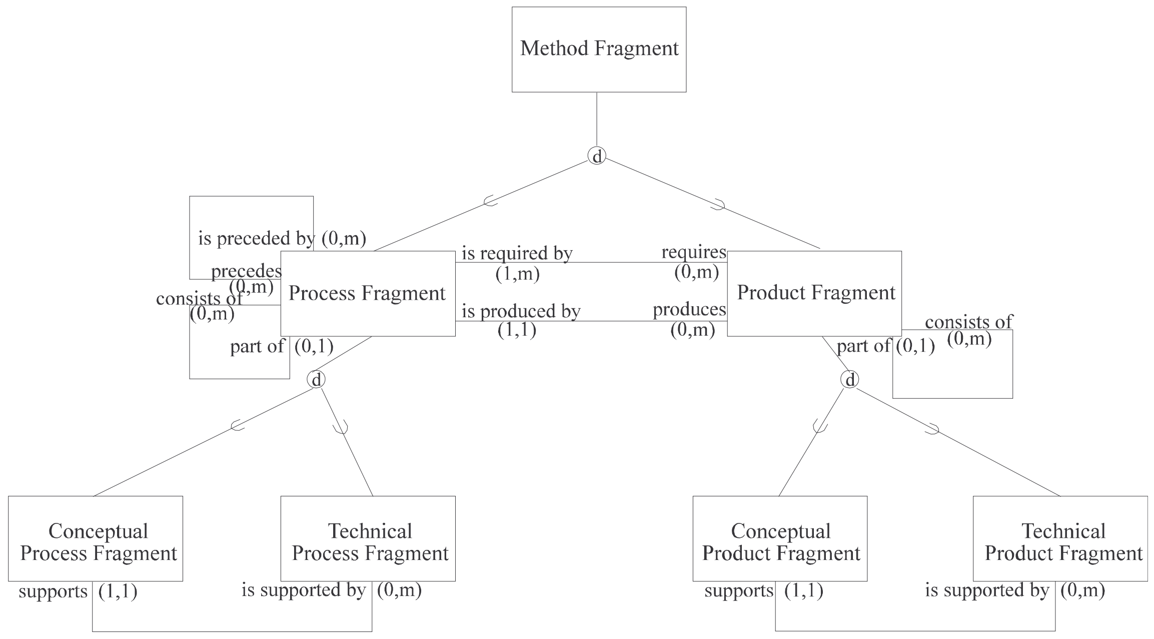


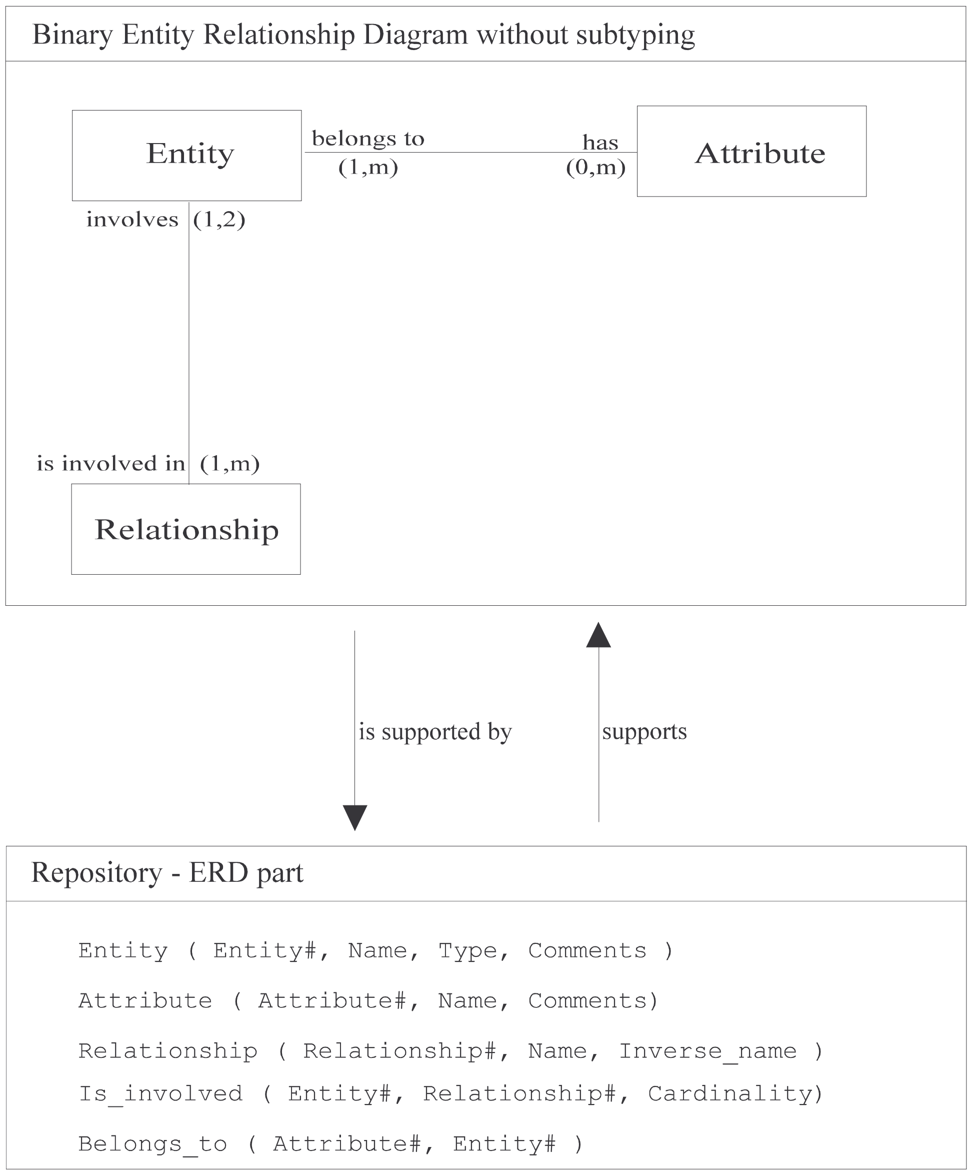
Figure 2.4 Data model of the method base

This data model contains several types of relationships:

- Relationships between fragments of the same type, which can be partitioned into:
  - relationships between granularity levels, such as the “Product fragment consists of Product fragment” relationship. An instance of this relationship is *Functional system design* consists of *Data model*, and
  - precedence relationships, which are only defined between process fragments, but which can be derived for product fragments. An instance of such a relationship is *Perform Information Planning* precedes *Perform Requirements Analysis*.
- Relationships between fragments of different types, which can be partitioned into:
  - support relationships, such as *Technical Product Fragment* supports *Conceptual Product Fragment*. These relationships express the fact that a conceptual method fragment, for instance an ERD meta model, is supported

by a technical method fragment, for instance the corresponding relational table code of the repository's ERD part (see figure 2.5).

- input/output relationships, expressing the fact that product fragments are required and are produced by process fragments. For instance, the process fragment “Construct data model” depicted in figure 2.3 produces an Entity Relationship Diagram, which is a product fragment.



```
Entity ( Entity#, Name, Type, Comments )
Attribute ( Attribute#, Name, Comments)
Relationship ( Relationship#, Name, Inverse_name )
Is_involved ( Entity#, Relationship#, Cardinality)
Belongs_to ( Attribute#, Entity# )
```

Figure 2.5 Example of a support relationship

These four types of relationships can be used to check different kind of consistencies in a situational method. This issue will be discussed in the next subsection.

In the complete data model, also properties of the concepts are included. Concepts and relationships to denote meta models of method fragments are considered as well. Moreover, relationships can be refined, such as the “produces” relationship that can be partitioned into “creates” and “updates” relationships. Examples of properties of a method fragment are: *goal*, *purpose*, *actors* responsible for the fragment, *complexity*, *amount of experience* needed to deal with it, and *resources*. Some concepts and relationships of, for instance, a meta process model can be: the concepts *task* and *trigger*, and the relationship *trigger is input of task*.

2.2. Method assembly with method fragments

In order to build situational methods, operations that manipulate method fragments are needed. These operations are facilities offered by the CAME tool. We consider three types of operations on method fragments:

- Administration operations, which are to be used to insert new fragments into the method base, to update fragments, and to delete fragments. An example of such an operation is: *Insert the Method/1 fragment “Complete Technical Design”*.
- Selection operations, which are used to retrieve method fragments from the method base. An example of such an operation is: *Retrieve all data base conversion fragments*.
- Assembly operations, which are used to configure the method fragments of the working set in order to assemble a situational method. An example of such an operation is: *combine fragment “Make data model for new database” with fragment “Complete Technical Design”*.

When assembling a situational method, several types of consistencies should be checked. These consistencies are derived from the four types of relationships identified in the data model of the method base.

The following types of consistency are distinguished:

- Consistency between granularity levels of the situational method. It should not be possible to combine fragments with very different granularity levels, for instance an activity “Identify data stores” preceded by a stage “Information Planning”. This type of consistency is very hard to check when fragments of different ISDMs are compared, because granularity levels are difficult to measure. In order to obtain a pragmatic solution to this problem, we have represented several ISDMs as trees. To each level of a tree, a number has been assigned, starting with the root. For instance, the sub-tree depicted in figure 2.6 consists of the levels 4, 5 and 6 of the ISDM Navigator [Navigator 91].

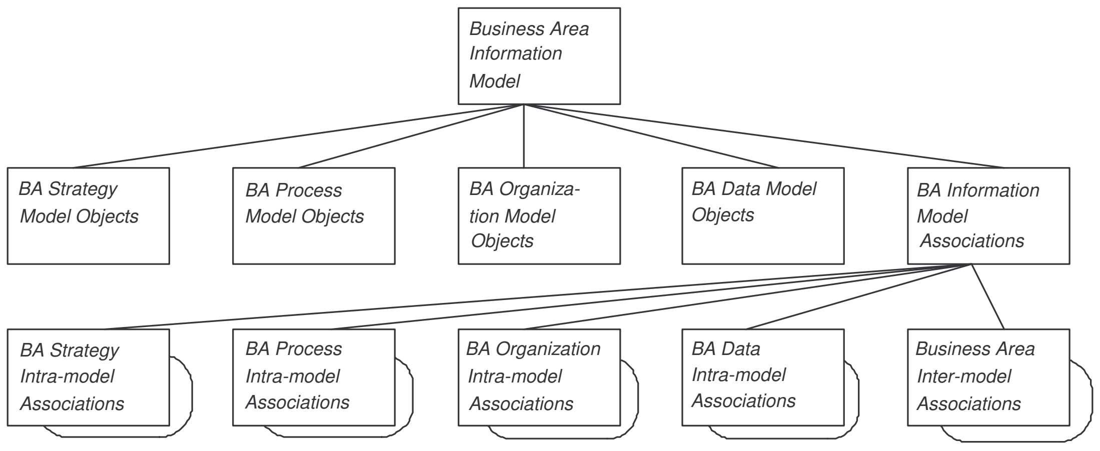

Business Area Information Model
BA Strategy Model Objects
BA Process Model Objects
BA Organiza-tion Model Objects
BA Data Model Objects
BA Information Model Associations
BA Strategy Intra-model Associations
BA Process Intra-model Associations
BA Organization Intra-model Associations
BA Data Intra-model Associations
Business Area Inter-model Associations

Figure 2.6Product fragment tree

The different levels of the tree correspond to granularity levels. Table 2.1 depicts examples of process fragments and product fragments of the granularity levels of Navigator.

	Process Fragment	**Product Fragment**
1	Perform Navigator	Converted ICPS
2	Analysis phase	Analysis deliverable
3	Business Area Requirements Analysis	Requirements Analysis report
4	Develop BAIM	Business area information model (BAIM)
5	Create BAIM associations	BAIM association
6	Create BA Inter-model associations	BA Inter-model association

Table 2.1 Granularity levels of Navigator

- Precedence consistency, i.e. consistency between consecutive stages of the situational method, including the products that are delivered consecutively.

For instance, the task “Inspect data model” of figure 2.3 should be preceded by a task that checks whether a data model is available. Also, this task requires a data model, which should be made available by a preceding task.

- Support consistency, i.e. consistency between conceptual and technical method fragments. Conceptual method fragments should be linked to the right technical method fragments and vice versa. It should not be possible that, for instance, an ERD is supported by a DFD editor.
- Input/output consistency, i.e. consistency between the data and the process perspective of a method. A product should always be produced by a corresponding activity. For instance, an ERD should not be defined as a product in a situational method without providing the ERD modelling procedure.

2.3. Formalisation of method fragments

In this sub-section we present a summary of the formalisation of method fragments [Harmsen 94]. This formalisation constitutes the basis of Situational Method Engineering consistency check rules.

In order to be able to reason about fragments, we introduce the following sets:

M : the set of method fragments

P : the set of process fragments

R : the set of product fragments

Recall from sub-section 2.1 that $M = P \cup R$, and that P and R are disjoint. Furthermore, we define

M_c : the set of conceptual method fragments

M_t : the set of technical method fragments

P_c, P_t, R_c, and R_t are defined similarly.

We also define:

V : the set of textual property values

$\mathbf{N}$: the set of natural numbers

It is assumed that for each relationship a predicate is defined that tests whether instances of fragments are involved in an instance of that relationship. The predicates of this kind we use in this sub-section are:

predicate consists_of **over** $M \times M$

predicate precedes **over** $P \times P$

predicate requires **over** $P \times R$

predicate produces **over** $P \times R$

predicate supports **over** $M_t \times M_c$

For instance, consists_of(x,y) is valid when for two given method fragments x and y, x consists of y. Note that product fragments can only consist of product fragments, and that process fragments can only consist of process fragments:

$$\text{consists\_of}(x,y) \Rightarrow (x \in P \wedge y \in P) \vee (x \in R \wedge y \in R)$$

We also presuppose for each property of a method fragment the availability of a function with which the value of that property can be retrieved. In this sub-section we use:

function goal **over** *M* **to** *V*

function level **over** *M* **to N**

For instance, level(*Business Area Information Model*) yields 4, which is the level of this product fragment (see sub-section 2.2).

The set of method outlines can be defined as a subset of *M*:

$$O \subseteq \{m \in M \mid \text{level}(m) \leq 3\}$$

Rule 1 and 2 are examples of consistency rules with respect to granularity levels. Rule 1 states that the level number of a fragment is one higher than the level number of its constituent parts:

$$\forall m_1, m_2 \in M[\text{consists\_of}(m_1, m_2) \rightarrow \text{level}(m_1) = \text{level}(m_2) + 1] \tag{1}$$

Rule 2 is a heuristic expressing that the levels of consecutive fragments in a method should not differ more than two:

$$\forall m_1, m_2 \in M[\text{precedes}(m_1, m_2) \rightarrow |\text{level}(m_1) - \text{level}(m_2)| < 3] \tag{2}$$

Rule 3 is an example of a precedence consistency rule. If a process fragment requires a product fragment, this should be produced by a preceding process fragment:

$$\forall p_2 \in P \forall r \in R \exists p_1 \in P[\text{requires}(p_2, r) \rightarrow \text{produces}(p_1, r) \wedge \text{precedes}(p_1, p_2)] \tag{3}$$

Note that the precedes relationship is transitive.

Rule 4 is an example of a support consistency rule, and states that a conceptual method fragment should always be supported by its corresponding technical method fragment. This requirement will be met if the *goals* of the fragments are the same:

$$\forall t \in M_t \forall c \in M_c[(\text{supports}(t, c) \rightarrow \text{goal}(t) = \text{goal}(c)] \tag{4}$$

Rule 5 is an example of an input/output consistency rule; it expresses the requirement that each product fragment should be produced by a process fragment:

$$\forall r \in R \exists p \in P[\text{produces}(p, r)] \tag{5}$$

3. APPROACHES TO SITUATIONAL METHOD ENGINEERING

In this section, we outline a spectrum in which several information systems project approaches are placed according to their degree of flexibility. Then we proceed with the description of several strategies aimed at the development of situational methods.

3.1.The Situational Method Spectrum

Considering the flexibility of methods, we distinguish in principle six approaches, varying from information systems development using a rigid standard method to a full use of Situational Method Engineering concepts. These six approaches can be placed in the so-called "Situational Method Spectrum", with which methods can be classified according to their degree of -controlled- flexibility.

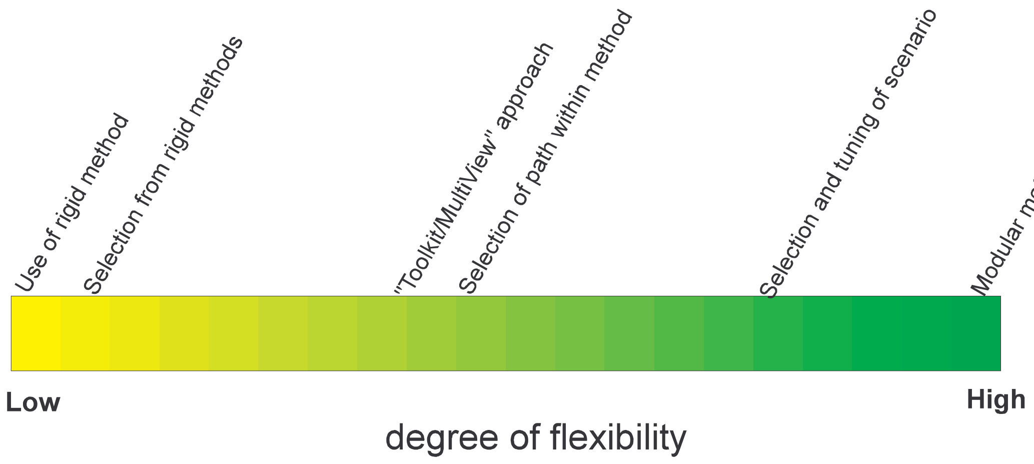

Use of rigid method
Selection from rigid methods
"Toolkit/MultiView" approach
Selection of path within method
Selection and tuning of scenario
Modular method construction
Low
High
degree of flexibility

Figure 3.1 The Situational Method Spectrum

To clarify each approach, we added to each description an example taken from the ICPS case.

- *Use of "rigid" methods*. A rigid method is a standard ISDM that inherently leaves no room for situational adaptation. Rigid methods adopt certain standards, based on one philosophy and suitable for one type of project. They consist of a fixed set of techniques, modelling procedures, and CASE tools. They provide no concrete guide-lines for adapting the ISDM to other project types.

  The adaptation of the inventory control system according to the standard ISDM Systems Engineering Methodology (SEM) employed by ABC is problematic. Like almost every standard ISDM, SEM aims at IS development, so the method has to be adapted by the project manager, which is time-consuming and completely intuition-driven. The quality of the resulting adapted method is doubtful.

- *Selection from rigid methods.* The second approach is the selection of one of the rigid methods, based on the project situation. After the choice of the most suitable ISDM for the project at hand, the method will be used without adaptations. Generally, method selection involves a lot of additional purchase and training costs [Kumar 92]. Moreover, it is still unlikely that the selected method meets all requirements of the project at hand.

  It turns out that there are two methods available aiming at information system adaptation. Both are, however, expensive, and the consultants of ABC are not used to particular exotic features. Moreover, both methods are too comprehensive for the ICPS project.

- *Toolkit/Multiview approach*. Another way to incorporate flexibility into information systems development is the strategy employed by the "Toolkit" [Benyon 87] or "MultiView" [Avison 90] approaches, which boils down to the inclusion of several methods, each addressing a specific aspect of the object system, into one method. It is dependent on the situation whether one of these "sub methods" will be employed in the project approach.

  Although the number of aspects covered would increase, it makes no sense for ABC to obtain a method based on the Toolkit/Multiview approach. Even if the method would consider information system adaptation, it is still too large for this project.

- *Paths within one method*. The fourth approach is the use of a method that enables to choose between several paths within the method. A well-known example of such a choice possibility is the difference being made in several methods between "classic" application development and "rapid" application development. More recently, there is a trend visible towards the inclusion of more paths, such as package selection, knowledge based systems development, and object-oriented development. An often used metaphor is the "Software Factory", in which the paths are represented by assembly lines.

  The choice of the method "Pilot", offering a number of paths, including information systems adaptation and data base conversion seems to be an *attractive option. Although some adaptation is still needed, such a path is relatively well tuned to the situation of the ICPS project. However, ABC develops also a lot of information systems with real-time aspects, and "Pilot" contains no path devoted to real-time information systems development.*

- *Selection and tuning of a method outline*. A further elaboration of this principle is the possibility not only to choose between different approaches, but also to tune a selected approach to the situation at hand. This involves the selection of a global method process model and data model, which will be further refined and adapted by the project manager. In order to do this efficiently, the method should be electronically available, and should be supported by method adaptation tools.

  The new state-of-the-art integrated CASE tool "AutoPilot" contains, besides an electronic version of "Pilot", also a method manipulation tool. With this tool, the project manager is able to adapt paths and to include new paths into the method. The workbench can therefore be employed for the ICPS project, but also for projects for which a path has not been defined yet.

- *Modular method construction*. The most radical solution, given the prerequisites mentioned before, is the modular construction of situational methods out of pre-defined building blocks, the method fragments. The fragments are available in one format, and are stored in a method base. They are assembled using rules that should guide the method engineer in constructing an effective, efficient, complete and consistent situational method.

  The new version of "AutoPilot" will be augmented with a method base. In this method base, not only descriptions of activity sequences are stored, but also descriptions of products. Moreover, -diagram- editors supporting the activities and repository descriptions are available, and are fully adaptable. Guidance and quality criteria are provided for the construction of situational methods

Note that the Situational Method Spectrum has to be interpreted with great care; it is obvious that flexibility is something that is very hard to measure. Note also that this spectrum depicts a chronological overview of the use of methods, starting with the advent of the use of standard methods, and ending with a situation that we envision to appear in large organisations by the end of this century.

The current situation concerning commercial methods is the following. Most of the modern commercial methods incorporate several aspects, most methods offer also a possibility to choose between two or more paths. Few methods are on-line available, offering controlled adaptation. There exist no commercial ISDMs that have adopted the modular method construction approach, although many large consultancy firms and software houses are doing research into this area. In our own research, we are focusing on both the selection and tuning of scenarios, or method outlines, and the modular construction of situational methods. This will be elaborated in the next sub-section.

3.2.Issues in Situational Method Engineering approaches

In section one, we described and graphically depicted the process of Situational Method Engineering. We identified the following steps: project characterisation, selection of method fragments, assembly of the fragments, validation of the situational method, and adaptation of the situational method. However, there are many issues related to these activities, some of which are described in this sub-section

1. Order of method engineering steps

The first issue concerns the order in which the method engineering steps should be gone through. Several alternatives can be distinguished, for instance:

a. First, the project is completely characterised, after which selection of method fragments based on the characterisation can take place;

   An inventory and risk analysis of the contingencies defining the ICPS project is made: IS adaptation, incorporation of standard software, database conversion, low complexity, average level of experience needed, costs less than 100'000 ECU, average response time < 1s, OS/2. After this, a selection of method fragments supporting these specific characteristics takes place.

b. First, the method fragments are selected, then the project is characterised. Selected fragments and project characterisation are matched, after which probably some adaptation of the set of selected fragments is needed.

   Relying on his experience, the ABC consultant selects the method fragments suitable for the ICPS project. After he has made his choice, he answers a number of questions asked by "AutoPilot". An example of such a question is: "How many entities contains the current data model?". The answers enable the tool to verify the choice of the consultant, and to propose possible improvements.

2. Uni-Method-Involvement versus Multi-Method-Involvement

The number of methods from which fragments can be selected is also an important variable in Situational Method Engineering. A distinction is made between the Uni Method Involvement (UMI) approach and the Multi Method

Involvement (MMI) approach [Harmsen 93]. In the UMI-approach, fragments from only one method can be selected, whereas in MMI, fragments from more than one method can be chosen. The latter approach enables a better matching between project situation and situational method, because the availability of more, carefully chosen methods increases the number of possible project situations to be covered. On the other hand, it makes the task of situational method assembly heavier, since all kinds of integrity and consistency problems between fragments of different methods can arise, see sub-section 2.3. Commercial CAME tools, such as Andersen Consulting's *Solution Configuration Tool* [Hidding 93] or Ernst & Young's *Automated Methods Environment* [Navigator 93], are all UMI-based.

> *Currently, the method base of "AutoPilot" consists solely of fragments from the "Pilot" method. This means that the situational method is configured according to a UMI approach. Since "Pilot" does contain fragments supporting IS adaptation and data base conversion, this seems no limitation. However, when ABC wants to use "AutoPilot" for real-time IS development, the method base should also consist of fragments from other methods, supporting this kind of development. This would result in an MMI approach.*

3. Product-oriented selection versus Process-oriented selection

The selection of method fragments can have at least two starting points: the products to be delivered, or the processes to be carried out. As product- and process fragments have relationships with each other, the selection of an instance of one type involves also the selection of instances of the other type. It depends on the situation and the taste of the method engineer whether the selection is product- or process-oriented. In practice, however, the selection will be neither completely product-, nor completely process-oriented.

> *For part of the adaptation, ABC recognises the following products, in our terminology: product fragments, to be delivered: a data model of the current system, a data model of the future system, a user interface design report and a database conversion plan. Given these products, the activities to be carried out and the tools to be used are –for the largest part– fixed, resulting in a situational method. If ABC had started with the definition of the activities: "First, we build data models of the current and the future system, then we write a data base conversion plan and then we design the user interface", this would have resulted in the definition of the products and tools.*

4. Degree of tool guidance

A high degree of tool guidance means that the CAME tool plays an active and "triggering" role in the Situational Method Engineering process. The method engineer provides a set of contingency factor values, after which the tool suggests a set of appropriate fragments. This set can be assembled into a situational method by the tool. The method engineer plays only a verifying and validating role. When the degree of tool guidance is low, the initiative is completely with the human, whereas the CAME tool plays only its role as a means to store, select and assemble fragments. The method engineer picks the fragments from the method base, he assembles them, after which he can, if he wishes, ask the tool to verify and validate the products he produced.

> *"AutoPilot" guides the user in an average manner. The method engineer can retrieve and assemble the method fragments he wishes to employ, after which the tool can validate his choice by asking questions about the situation and by applying consistency rules to the constructed situational method. If the tool was more guiding, the process of selecting and assembling method fragments would have been more automated and would have involved more AI techniques.*

5. Top-down or bottom-up method assembly

Another point of interest is the question whether situational methods should be constructed using a method outline or by building block-by-block. The first type we call *top-down* assembly, whereas the latter type is called *bottom-up* assembly. If a situational method is constructed top-down, the task of selecting method fragments is split into two sub tasks: first, a method outline is selected, then a number of method fragments is chosen. The choice of a specific method outline restricts the number of possible method fragments to be inserted in the situational method, and therefore simplifies the task of method assembly.

> *ABC chooses to proceed top-down. First a Package Selection method outline and an IS Adaptation method outline are chosen and combined. Quite a lot superfluous product and process fragments are deleted, and the resulting method is checked by "AutoPilot". Some suggestions are taken for granted, others not; a few fragments have to be added. After a second consistency check, "AutoPilot" generates the project repository and the situational method.*

4. COMPUTER AIDED METHOD ENGINEERING

Since designing a method is, like information systems design, an information creation and transformation process, Situational Method Engineering has a lot in common with Information Systems Engineering. Similarities do not stop at the computer aided support for the activities involved. Due to its complex nature, Situational Method Engineering should always be applied in conjunction with supportive tools. Just as the growing complexity of information systems and the need for quality and maintainability caused the advent of Computer Aided Systems Engineering (CASE) technology [McClure 89], the need for consistent, effective and maintainable methods justifies Computer Aided Method Engineering (CAME).

The main difference of CAME with respect to CASE constitutes the domain they support. CASE applies to real-world apllication domains, CAME uses the domain of information systems development methods. Just as CASE is often being designated as "the automation of the automation", CAME could be described as: "the automation of automation methods assembly".

In this section we describe the requirements for a CAME tool. We provide also a CAME achitecture, with which our prototype complies.

4.1.CAME Tool requirements

An effective CAME tool should provide support for the following method engineering activities:

- *Determination and valuation of contingency factors*

  The tool should provide an interface that enables the method engineer to determine contingency factor values. The contingency factors have to be classified according to certain rules. The tool also has to recognise and to weigh mutual dependencies between contingency factors, that is, the constraining impact of one contingency factor upon others. For instance, the contingency factor OS/2 affects other factors, since it requires a certain amount of internal and external memory, having an effect on the costs.

- *Storage of method fragments*

  In order to be able to manipulate method fragments, they have to be available to the CAME tool. Therefore, a method base should be provided, from which method engineers can select the most appropriate fragments, including method outlines. As experience accumulates and new methods arise, the method base has to be modified from time to time. New method fragments have to be added, or fragments have to be modified or deleted. These are the methods administration tasks.

- *Retrieval and composition of method fragments*

  The method engineer should be able to select method fragments from the method base. Having selected the fragments, manipulation of them is needed. Facilities of this type include deleting and moving method fragments within the method under construction.

Composing a situational method consists of, among other things, matching the project characterisation with the method fragment descriptions. A CAME tool should therefore incorporate functionality to support this matching process, for instance by offering views on a -limited- set of method fragments that match to a certain extent the project characterisation.

Situational methods and their supporting tools are integrated. Therefore, the CASE tool to be used should be fully adaptable. Currently, such adaptable CASE tools, designated by the term CASE shell or meta-CASE tool, are available. Examples of such tools are RAMATIC [Bergsten 89], MetaEdit [Smolander 91], ToolBuilder [Alderson 91] and Maestro II [Merbeth 91]. It is of importance that the CAME tool generates output that can be used by the meta-CASE tool.

- *Validation and verification of the situational method*

  The tool should not only offer support for the selection and assembly of method fragments, but should also incorporate guide-lines to ensure that a sufficient and complete set of fragments has been chosen and that it has been assembled in a consistent way.

- *Adaptation of the situational method*

  The CAME tool should offer functionality to support this dynamic method-adaptation. When the project is finished, the results of experiences with the scenario should be accumulated in the method base, in order to be able to assimilate practical experience into future situational methods.

4.2.Architecture of a CAME Tool

In order to meet the requirements we outlined in sub-section 4.1, we are currently developing a CAME tool, called Decamerone. This CAME tool is based on and is used in conjunction with the meta-CASE tool Maestro II [Merbeth 91]. The architecture of Decamerone is depicted in figure 4.1.

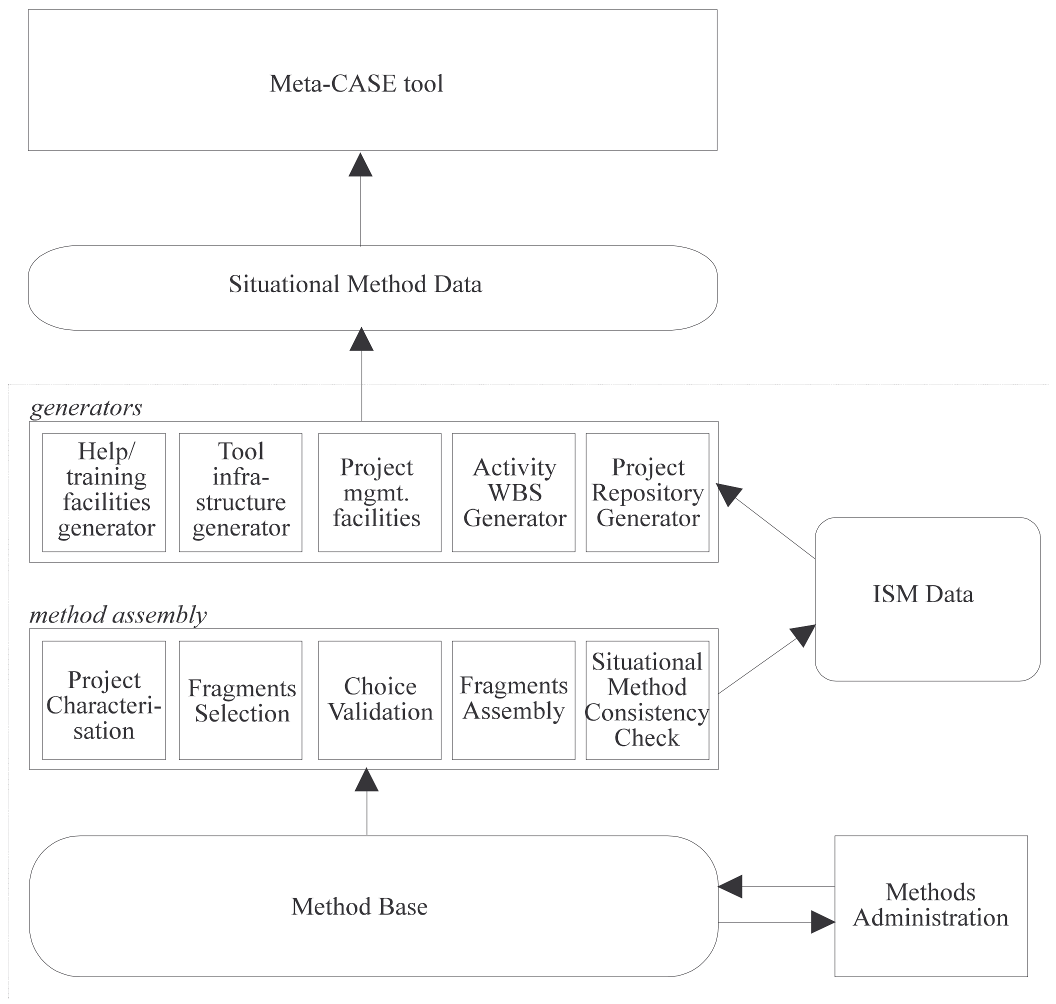
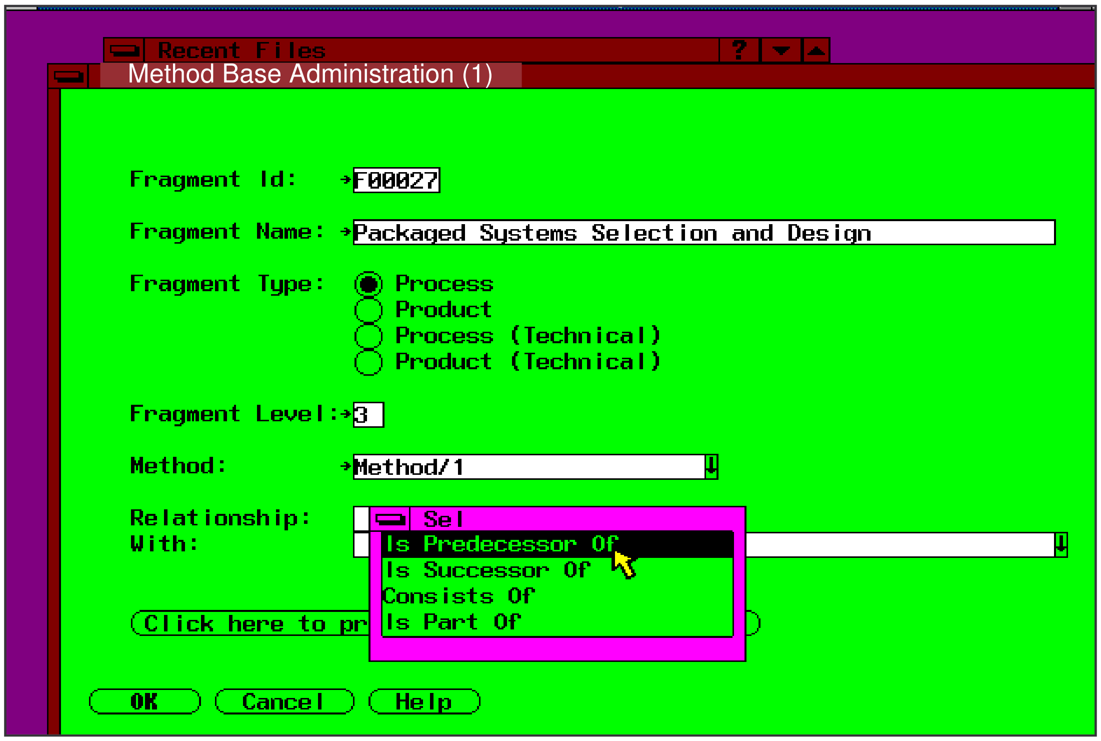


Figure 4.1 Architecture of a CAME tool

Arrows in this figure depict data flows, rounded rectangles depict data stores and boxes depict Method Engineering support functionality. Method fragments, including method outlines, are stored in the method base. Fragments are inserted, modified, or deleted through the methods administration component. The method assembly component is used for the selection and assembly of fragments from the method base. It provides also an interface through which the project situation is characterized, and it provides method consistency check facilities. The method assembly component results are stored in a CAME-internal ISM (Intermediate Situational Method) data base. This Intermediate Situational Method is processed by several generators, which provide the meta-CASE tool with the Situational Method data. These data include a project repository, a work-breakdown structure, project planning data, a set of diagram- and text editors, and help screens and paper manuals. The meta-CASE tool needs this input in order to fulfil its role as an integrated CASE tool during the IS project.

Figure 4.2 shows the method fragment insertion screen, one of the methods administration interface screens of the Decamerone prototype. Besides the name of the method fragment, the methods administrator specifies a unique fragment identifier. This is because different method fragments can have the same name. Furthermore, the *type*, and the *granularity level* of the fragment (see section 2) are specified, as well as its relationships with fragments already stored in the method base. He also indicates the types of project in which the fragment can be used. Similar screens are used to specify the fragments's contingency factor values, and to perform other operations.



Fragment Id: F00027

Fragment Name: Packaged Systems Selection and Design

Fragment Type: [✓ Process] Product Process (Technical) Product (Technical)

Fragment Level: 3

Method: Method/1

Relationship:
With:

Figure 4.2 Methods administration interface of Decamerone

5. CONCLUSIONS AND FURTHER RESEARCH

From the necessity of controlled flexibility in systems development organisations we have introduced the concept of engineering of situational methods. Configuration of methods based on proven parts of methods, the so-called method fragments, allow to standardise in notations and tool support without enforcing rigid work procedures on the whole of creative systems development. The method configuration process, i.e. selection and assembly of method fragments, is supported by a Computer Aided Method Engineering (CAME) tool. A prototype of a CAME tool is developed, containing functionality for method configuration, method administration, and generating input for the metaCASE environment. The method base, being the central repository of method fragments in the CAME tool, is structured according to the meta-models of process fragments and product fragments.

Methods in paper manuals are passé. Automated methodical support, also called on-line methods or electronic handbooks [Heym 93], allow for situational tuning to the specific project circumstances. Furthermore, CAME tools can generate exactly tailor-made methods and tools for each of the members of the project team. Developers need not to be overloaded with the complete method or tool, but can be equiped solely with support for their part of the project.

This paper on Situational Method Engineering is written in the context of the large research program on method engineering at the University of Twente. Various studies involving meta-modelling of methods and CASE tools are currently being carried out. The structure and contents of the method base need further refinement by means of formalisation and tests on all kinds of methods, preferably from diverse sources and with different underlying paradigms. Conventional methods, object oriented methods, method for knowledge based systems, or for real-time systems, all candidates to be inserted into the method base. An additional problem to tackle are the ways to fragment a method.

The CAME tool will be developed in the coming years and tested in the systems development practise. The support for method configuration will be expanded, taking advantage of Artificial Intelligence experiences. Empirical research into the practice of method building and method tuning is needed to obtain more consistency rules. Consistency rules will be formalised as complete as possible, in order to allow for an easy incorporation into the CAME tool.

Right now, the CAME tool is based on the meta-CASE tool Maestro II. In the future, we will provide links to other meta-CASE tools. This will have its impact on the current architecture, which will be divided in a meta-CASE independent and a meta-CASE dependent component. However, this not a research topic with top priority.

REFERENCES

[Alderson 91] Alderson, A., Beyond today's CASE technology towards Meta-CASE. In: Proceedings of the third European CASE Conference, London, 1991.

[Avison 88] Avison, D.E., and G. Fitzgerald, Information Systems Development: Methodologies, Techniques and Tools, Blackwell Scientific Publications, Oxford, 1988.

[Avison 90] Avison, D.E., and A.T. Wood-Harper, Multiview: an exploration in information systems development, Blackwell Scientific Publications, Oxford, 1990.

[Batini 92] Batini, C., S. Ceri, S. Navathe, Conceptual Database Design: An entity-relationship approach, The Benjamin/Cummings Publishing Company, Redwood City (CA), 1992.

[Benyon 87] Benyon, D., and Skidmore, S., Toward a Tool-Kit for the Systems Analyst. In: The Computer Journal, Vol. 30, No. 1, 1987.

[Bergsten 89] Bergsten, P., J. Bubenko, R. Dahl, M. Gustafsson, L.-Å Johansson, RAMATIC - A CASE Shell for Implementation of Specific CASE Tools, TEMPORA T6.1 Report, SISU, Stockholm, 1989.

[Bots 87] Bots, P.W.G., and H.G. Sol, An environment to support problem solving. In: Decision support systems, vol. 3, no. 3, pp. 225-231, 1987.

[Brinkkemper 90] Brinkkemper, S., Formalisation of Information Systems Modelling, Dissertation University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

[Coad 90] Coad, P., and E. Yourdon, Object-Oriented Analysis, Prentice-Hall, Englewood Cliffs, 1990.

[Euromethod 93] Euromethod Architecture, Euromethod project deliverable Work Package 2, 1993.

[Finkelstein 92] Finkelstein, A., J. Kramer, B. Nuseibeh, Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. In: International Journal of Software Engineering and Knowledge Engineering, Vol. 2, No. 1, pp. 31-57, 1992.

[Harmsen 93] Harmsen, F., and S. Brinkkemper, Computer Aided Method Engineering based on Existing Meta-CASE Technology. In: Brinkkemper, S. and F. Harmsen (Eds.), Proceedings of the

Fourth Workshop on The Next Generation of CASE Tools, Paris, 1993.

[Harmsen 94] Harmsen, F., S. Brinkkemper, H. Oei, Formalisation of Method Fragments, in preparation.

[Heym 93] Heym, M., and H. Österle, Computer-aided methodology engineering. In: Information and Software Technology, vol.35, no. 6/7, pp. 345-354, 1993.

[Hidding 93] Hidding, G.J., G.M. Freund, J.K. Joseph, Modeling Large Processes with Task Packages. Workshop on Modeling in the Large, AAAI Conference, Washington, D.C., 1993.

[Hong 93] Hong, S., G. van den Goor, S. Brinkkemper, A Comparison of Object-Oriented Analysis and Design Methodologies. In: Proceedings of the 26th Hawaiian Conference on System Sciences (HICSS-26), IEEE Computer Science Press, 1993.

[Kumar 92] Kumar, K., and R.J. Welke, Methodology Engineering: A Proposal for Situation-Specific Methodology Construction. In: W.W. Cotterman, J.A. Senn (Eds.), Challenges and Strategies for Research in Systems Development, Wiley, 1992.

[Lindgreen 90] Lindgreen, P. (Ed.), A Framework of Information Systems Concepts. Interim Report of the IFIP WG8.1 Task Group FRISCO, 1990.

[Marttiin 92] Marttiin, P., K. Lyytinen, M. Rossi, V.-P. Tahvanainen, K. Smolander, J.-P. Tolvanen, Modeling Requirements for Future CASE: Issues and Implementation Considerations. In: DeGross, J.I., J.D. Becker, J.J. Elam (Eds.) Proceedings of the 13th ICIS, Dallas, 1992.

[Marttiin 93] Marttiin, P., M. Rossi, V.-P. Tahvainanen, K. Lyytinen, A Comparative Review of CASE Shells - a preliminary framework and research outcomes. In: Information and Management, vol.25, no.1, pp 11-31, 1993.

[Maynard 39] Maynard, H.B., and G.J. Stegemerten, Operation Analysis, McGraw-Hill, New York, 1939.

[McClure 89] McClure, C.L., CASE is Software Automation, Prentice-Hall, Englewood Cliffs, 1989.

[Merbeth 91] Merbeth, G., Maestro II - the integrated CASE system of Softlab (in German: Maestro II - das integrierte CASE-System von Softlab). In: Balzert, H. (Ed.), CASE Systeme und Werkzeuge, 3e Auflage, BI Wissenschaftsverlag, 1991.

[Method/1 92] Method/1® version 9.0 methodology manuals, Andersen Consulting, 1992.

[Navigator 91] NAVIGATORsm system series, release 1.5, Ernst & Young, 1991.

[Navigator 93] NAVIGATORsm system series, release 2.0, Automated Methods Environment, Ernst & Young, 1993.

[Nuseibeh 92] Nuseibeh, B., and A. Finkelstein, ViewPoints: A Vehicle for Method and Tool Integration. In: Proceedings of the International Workshop on CASE (CASE 92), Montreal, IEEE Computer Society Press, pp. 50-60, 1992.

[Oei 92] Oei, J.L.H., L.J.G.T. van Hemmen, E.D. Falkenberg, S. Brinkkemper, The Meta Model Hierarchy: A Framework for Information Systems Concepts and Techniques, Technical Report No. 92-17, Department of Informatics, University of Nijmegen, 1992.

[Oei 94] Oei, J.L.H., and E.D. Falkenberg, Harmonisation of Information System Modelling and Specification Techniques. submitted for publication, 1994.

[Olle 88] Olle, T.W., Business Analysis and System Design Specifications for an Inventory Control and Purchasing System. In: Olle, T.W., A.A. Verrijn-Stuart, L. Bhabuta (Eds.), Computerized Assistance during the Information Systems Life Cycle, North-Holland, 1988.

[Olle 91] Olle, T.W., J. Hagelstein, I.G. MacDonald, C. Rolland, H.G. Sol, F.J.M. van Assche, A.A. Verrijn-Stuart, Information Systems Methodologies - A Framework for Understanding, 2nd edition, Addison-Wesley, 1991.

[Partsch 90] Partsch, H.A, Specification and Transformation of Programs, Springer Verlag, 1990.

[Rumbaugh 91] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-oriented modeling and design, Prentice-Hall, Englewood Cliffs, 1991.

[Saeki 93] Saeki, M., K. Iguchi, K. Wen-Yin, M. Shinohara, A Meta-Model for Representing Software Specification & Design Methods. In: N. Prakash, C. Rolland, B. Pernici (Eds.), Proceedings of the IFIP WG8.1 Conference on Information Systems Development Process, Como, 1993.

[Slooten 93] Slooten, K. van, and S. Brinkkemper , A Method Engineering Approach to Information Systems Development. In: N. Prakash, C. Rolland, B. Pernici (Eds.), Proceedings of the

IFIP WG8.1 Conference on Information Systems Development Process, Como, 1993.

[Smolander 91] Smolander, K., P. Marttiin, K. Lyytinen, V.-P. Tahvainanen, Meta-Edit - A Flexible Graphical Environment for Methodology Modelling. In: R. Andersen, J.A. Bubenko, A. Solvberg (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science 498, Springer, Berlin, 1991.

[Verhoef 91] Verhoef, T.F., A.H.M. ter Hofstede, G.M. Wijers, Structuring Modelling Knowledge for CASE shells. In: R. Andersen, J.A. Bubenko, A. Solvberg (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science 498, Springer, Berlin, 1991.

[Vessey 92] Vessey, I., S. Jarvenpaa, N. Tractinsky, Evaluation of Vendor Products: CASE tools as Methodology Companions. In: Communications of the ACM, vol. 35, no. 4, pp 90-105, 1992.

[Wijers 91] Wijers, G.M., Modelling support in information systems development, Dissertation University of Delft, Thesis Publishers, Amsterdam, 1991.

[Wirfs-Brock 90] Wirfs-Brock, R., B. Wilkerson, L. Wiener, Designing object-oriented software, Prentice-Hall, Englewood Cliffs, 1990.

[Yourdon 89] Yourdon, E., Modern Structured Analysis, Prentice-Hall, Englewood Cliffs, 1989.

View publication stats