CREWS Report Series 97-06

A PRIMER FOR METHOD ENGINEERING

Colette Rolland

Université de Paris 1 - Sorbonne
17, rue de la Sorbonne
75231 Paris Cedex 05
Rolland@masi.ibp.fr

Proceedings of the conferance INFORSID
(INFormatique des ORganisations et Systèmes d'Information et de Décision)
Toulouse, France, June 10-13, 1997

A PRIMER FOR METHOD ENGINEERING

Colette Rolland

Université de Paris 1 - Sorbonne
17, rue de la Sorbonne
75231 Paris Cedex 05
Rolland@masi.ibp.fr

Résumé

L'ingénierie des méthodes répond à l'incapacité des méthodes à prendre en compte les besoins de leurs utilisateurs, les ingénieurs d'application. L'ingénierie des méthodes a pour objectif d'aider à la construction de méthodes. L'article utilise le cadre de référence des quatre mondes pour étudier le sujet de l'ingénierie des méthodes. Il présente un état de l'art qui identifie les problèmes importants et les directions de recherche à suivre pour les résoudre. L'article met également l'accent sur les problèmes majeurs qui n'ont pas été encore abordés ou qui sont apparus récemment.

Mots-clés

Ingénierie des méthodes, ingénierie des processus, fragment de méthode, méthode situationnelle

Abstract

The area of method engineering has emerged in response to an increasing feeling that methods are not well suited to the needs of their users, the application engineers. Method engineering aims at developing methods. The paper uses the four worlds framework to investigate the subject of method engineering. It presents a state-of-art survey with a view to identifying the important problems and research directions being followed to solve these. The major problems not yet addressed or which have newly emerged are highlighted

Key-words

Method engineering, process engineering, fragment method, situational method

1. Introduction

There are a large number of methods available for information systems development (ISD). These include structured approaches, prototyping approaches, systemic approaches, object-oriented, etc. Many of these methods have been comparatively analysed in books [OLLE88] and journals (e.g [JACK84], [HIRS92]). Despite a large body of work concerning details of systems development methods, there is still a poor understanding of how such methods are actually used in practice [WYNE93]. However, there is an increasing feeling that methods are not well-suited [LYYT87] to the needs of their users, the IS developers. In particular, it is necessary to change methods from one business situation [HIDD94] to another. Several survey based studies (e.g [WIJE90], [AAEN92], [YOUR92], [RUSS95] have revealed that ISD methods are developed or adapted locally. For example, a survey of method use in over 100 organisations' [RUSS95] shows that more than 2/3 of the companies have developed or adapted their methods in-house. Also, 89% of respondents believed that methods should be adapted on a project to project basis.

Method engineering [WELK92a] represents the effort to improve the usefulness of systems development methods by creating an adaptation framework whereby methods are created to match specific organisational situations. There are at least two objectives that can be associated to this adaptation. The first objective is the *production of contingency methods,* that is, situation-specific methods for certain types of organisational settings. This objective represents method engineering as the creation of a multiple choice setting. The second objective is one in which method engineering is used to produce *method "on-the-fly"*. Situational method engineering [WELK92a] is the construction of methods which are tuned to specific situations of development projects. Each system development starts then, with a method definition phase where the development method is constructed on the spot.

Another broad view of the method engineering area is proposed by Harmsen who suggests to organise approaches to method engineering in a Method Spectrum [HARM94] according to the degree of flexibility in meeting situational needs. Methods are placed on a scale ranging from 'low' flexibility to 'high' (figure 1). At the 'low' end of this spectrum are rigid methods whereas at the 'high' end is modular method construction. *Rigid methods* are completely pre-defined and leave little scope for adapting them to the situation at hand. On the other hand, *modular methods* can be modified and augmented to fit a given situation. *Selection from rigid methods* allows each project to choose its method from a panel of rigid, pre-defined methods whereas *selection of paths within a method* consists of selecting the appropriate path for the situation at hand. Finally selection and tuning a method permits each project to select methods from different approaches and tune them to the project's needs.

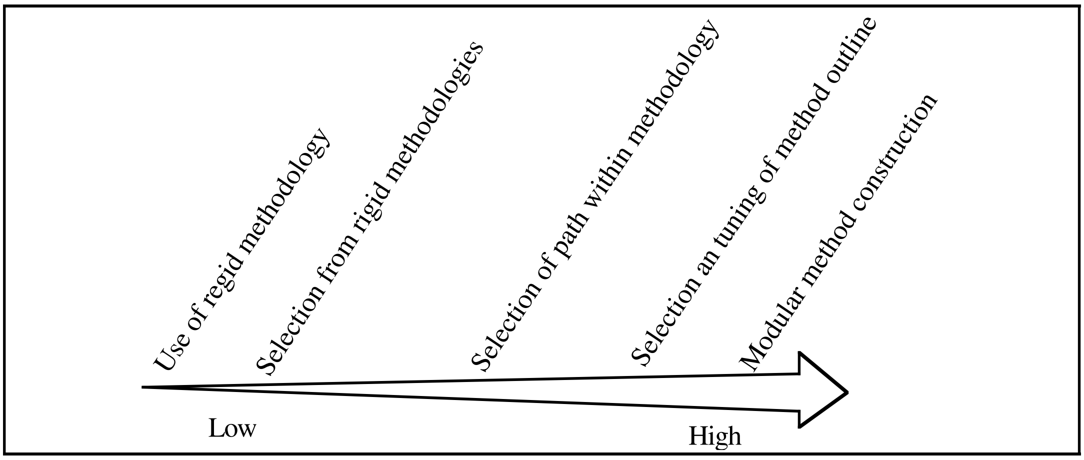

Figure 1: the spectrum of ME approaches

2. A framework for understanding method engineering

The position taken in this paper consists of adapting the *four worlds framework* originally proposed for system engineering (figure 2) to method engineering. The framework has proved its efficiency to enhance understanding in various engineering disciplines, namely information systems engineering [JARK92], requirements engineering [JARK93] and IS development process engineering [ROLL97a]. Our claim is that it can help understanding the field of method engineering which consists of applying engineering approaches, techniques, and tools to the construction of methods.

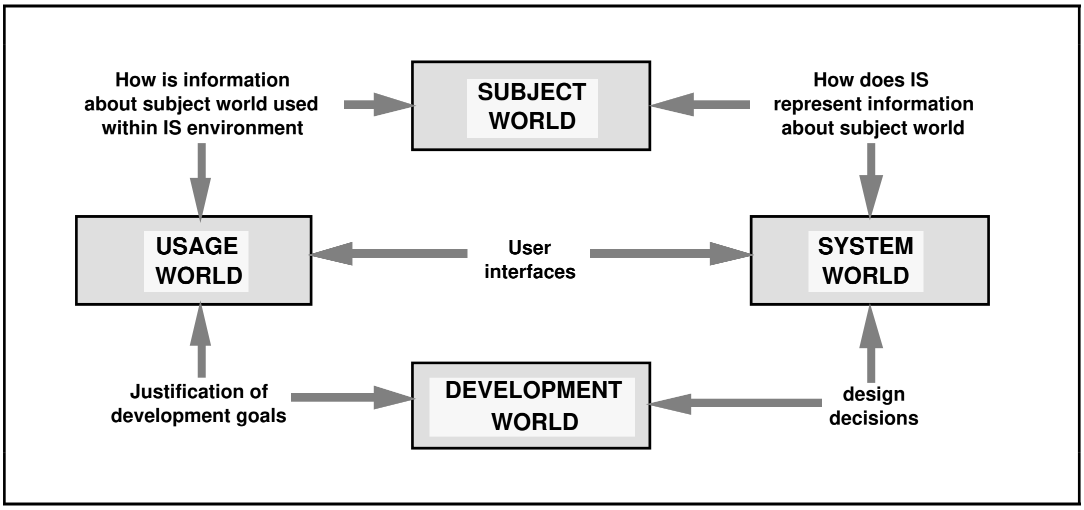

Figure 2 : The four worlds of IS engineering

In the original system engineering framework (figure 2), the *subject world* contains knowledge of the domain about which the proposed IS has to provide information. It contains real-world objects which become the subject matter for system modelling.

The *system world* includes specifications at different levels of detail of what the system does. It holds the modelled entities, events, processes, etc. of the subject world together with the mapping of these conceptual specifications onto design specifications and implementation.

The *usage world* describes the organisational environment of the information system, i.e. the activity of agents and how the system is used to achieve work, including the stakeholders who are system owners and users. The usage world deals with the intentional aspects of the IS to be build whereas the subject world refers to the domain it shall represent through informations.

The *development world* focuses on the entities and activities which arise as part of the engineering process itself. It contains the processes which create the information system i.e. processes which involve analysis and understanding of knowledge contained in the other worlds and representation of that knowledge.

Our proposal consists of identifying the *subject world* to the *world of methods*. Both Oxford and Webster's dictionaries primarily define the term "method" as meaning "the procedure for obtaining an object". Method is clearly a concept of process rather than representation, even the current methods in use focussed on representations more than on the process to get these representations. This paper will avoid the term *methodology* as its original meaning (study of methods) has become confused, and is either used as a simple synonym of method (cf [OLLE88]) or to create a hierarchy of methods [WYNE93]. One shall notice that a method is an "artifact", i.e an object made by people, for subsequent use. The use of the method is itself to create artifacts (information systems) for subsequent use. An artifact has a physical persistence and reflects the the time period and cultural stage it was made.

The *system world* deals with the *representation of methods*. As methods are artifacts to create artifacts in order to process real things, the framework is lifting the systems structures to a higher (third) level of abstraction. This level is often referred to as the meta-modelling level.

In the *usage world* we will investigate the reasons, the *rationale for method engineering*. We shall respond to the WHY question. Why an introduction of the third level should resolve problems which were not solved with the two other levels? Clearly the hope is that with the introduction of the third level, the actual development methods become "selectable" (or definable), and equally importantly, the determination of these selections itself becomes more higly structured.

The *development world* deals with the *process of constructing methods*, either "on-the-fly" or by "selection". It is a meta-process in the sense that it supports the construction of methods, which in turn, will support the development of information systems. The way the process might be supported by a tool environment (Computer Aided Method Engineering i.e CAME tool) is also a concern of this world.

The paper uses the four worlds to present the state of art in method engineering and to raise questions, problems and research issues in the field of method engineering. It comprises four sections, each of these relating to one of the world.This allows us to discuss in a focussed manner the different concerns of method engineering : the *definitions of methods,* their *representations*, the *way of developing these representations*, and *the rationale for using these representations*.This is done in the subject, system, development, and usage worlds respectively.

3. The subject world

The subject world is concerned with the *definition of methods.* Many definitions have been proposed [BRIN90], [PRAK94], [WYNE93], [LYYT89] and most of them converge to the idea that a method is based on models (systems of concepts) and consists of a number of steps which must/should be executed in a given order. Seligmann [SELI89] proposes a framework for *methods* which comprises the *way of thinking* (the philosophy), the *way of modelling* (the models to be constructed), the *way of organising* and the *way of supporting*. The *way of organising* is subdivided into the *way of working*, i. e. the how of performing the development, and the *way of control*, i. e. the how of managing the development. The *way of supporting* deals with the description techniques and the corresponding tools).

In the past, method developers have concentrated on the definition of models more than on the definition of guidelines. This means that the *product aspect* of method has been favoured at the expense of the *process aspect.* Several classifications of product models have been proposed. A traditional distinction regarding the modelling perspectives is between the *structural, functional* and *behavioural* perspective [OLLE88]. Recent developments of methods emphasise the need for enterprise modelling [BUBE94] and introduce concepts such as goal, actor, role, role dependency and the like [YU94], [DARD91], [POTT94]. At the same time, method developers introduced the distinction between functional and non functional requirements for information systems development. This leads to a broader classification of models according to the *functional* (itself subdivided into structural, functional and behavioural), *non functional* , and *intentional* perspective [ROLL97b].

It is clear that process modelling has been paid less attention than product modelling. Recent in-depth studies of software development practices [LUBA93] demonstrate that we know very little about the development process. However there has been recently a shift of focus from the product to the process view of systems development and process engineering is considered today as a key issue by both the software engineering comunity and the information systems engineering community. There is already considerable evidence for believing that there shall be both, improved productivity of the software systems industry and improved systems quality, as a result of improved development processes[DOWS93], [ARME93], [JARK94].

According to Dowson [DOWS88], process models can be classified in three groups: activity-oriented models, product-oriented models and decision-oriented models (Figure 3). As can be seen in Figure 3, there has been a remarkable surge of interest in process modelling in the last ten years after a gap of nearly 30 years of stability. *Activity-oriented process models* [ROYC70] come from an analogy with problem-solving and provide a frame for manual management of projects. This linear view is inadequate for methods which support backtracking, reuse of previous designs, and parallel engineering.However they have been predominent for the last twenty years. Current *workflow* and *software* process models [OSTE87], [FINK94] are extensions of activity-based models but they assume broad coverage and fairly strict control of the process.

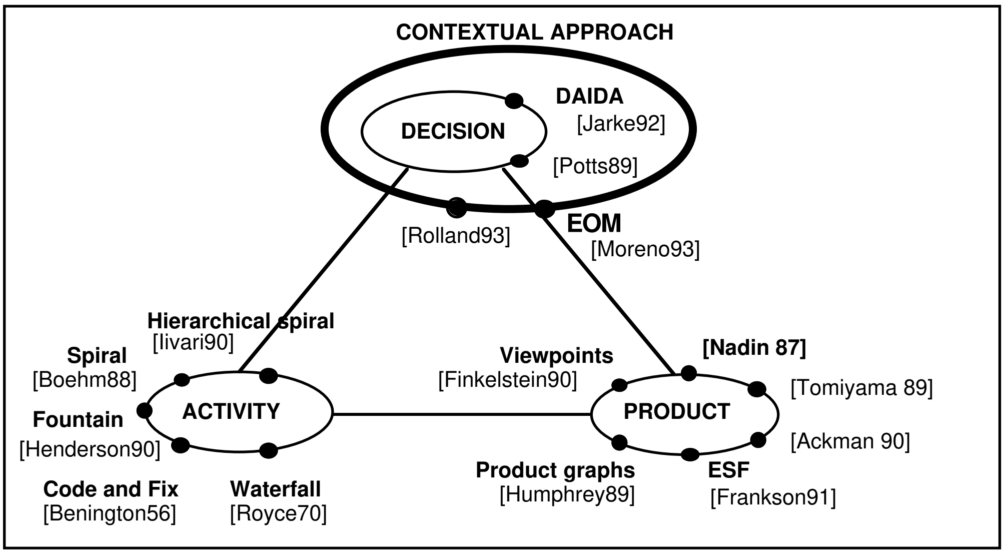

Figure 3: Classes of Process Modelling Approaches

Product-oriented process models [HUMP89], [FINK90], [FRAN91] represent the development process through the evolution of the product. They permit design tracing in terms of the performed transformations and their resulting products. Finally, the most recent models are *decision-oriented models.* They integrate more deeply the semantics attached to evolutionary aspects. The notion of design decision facilitates understanding of the designer's intention, and thus better reuse of results [Potts89]. The situatedness of engineering processes has been taken into account and coupled to a decision-based approach in the *contextual approach* for requirements engineering developed in the NATURE project [ROLL91], [ROLL93], [ROLL94b], [ROLL95].

Methods cannot be separated from the technology available for supporting their use i.e from Computer Aided System Engineering (CASE) technology. Reflecting the bias in favour of the product aspects of methods, most existing CASE tools are primarily providing facilities to capture, store, manipulate and document products and rarely support the process of developing products. Process support has recently been introduced through enactment mechanisms in Software Process Centred Environments (see Finkelstein 1994 for a complete survey), workflow systems, and a new range of CASE tools like [SISA96].

Improving process models and supporting their enactment is a major research issue. Reducing the "YAMA (Yet Another Modelling Approach) syndrome" [OIE92] is surely a driving force for method engineering. There is a need for a systematic and objective means to compare and evaluate methods.

4. The system world

The system world is concerned with the *representation of methods.* Based on our assumption that a method is composed of a product model and a process model, the representation of methods really deals with modelling models i.e meta-modelling. Some work has been done on the static integration of product models. Venable [VENA93] has performed detailed analyses and integrations of both data flow models and conceptual data models. [CAMP94] have analysed levels of abstraction for conceptual schemas. [HONG93] compare eight object-oriented methods on the basis of their data models. A "super method" acts as a reference model for the methods compared. [IIVA94], in accordance with the CRIS approach [OLLE88], relies on a normative comparison of object-oriented methods, as he draws up a number of requirements which are compared to the methods' properties. [WIER91] has compared JSD, ER modelling and DFD modelling. [FALK94] have proposed to integrate models in a metamodel hierarchy in which models are related one with another through transformations.

Even if these works are of interest for the method engineering community, they do not entirely fulfil the objective of methods representation as stated in ME. Method engineering aims at either facilitating the selection of a method within a panel of contingency methods or to support the selection and the assembly of method components to construct a method on-the-fly. Therefore the representation of methods is purposeful and must satisfy two requirements : (1) identify the right notion of a method component and (2) characterise a component to facilitate its situated usage. Complementarily the question of the language that can be chosen for representation prurpose is raised. We comment on these three points in turn.

Meta-modelling languages

Brinkkemper [BRIN90] claims that every conceptual modelling language is suitable to serve as a meta-modelling language, and therefore suitable to represent method components. Various meta-modelling applications of languages originally intended for other domains, such as LOTOS [SAEK91] seem to demonstrate the validity of this claim. However other authors have opposite claims. Venable and Grundy claim that there is a need for a specific meta-modelling language and environment. CoCoa [Vena93] has been developed and used in the MViews environment [GRUN96].

Some efforts have been made to define general requirements for ME : [KOTT84], [MART95], and [WELK92a] recognise the need for rich semantic constructs for modelling the conceptual structure and constraints of methods. [ROLL95] present a set of characteristics of process models which a process meta-model should cope with.

Most of the meta-modeling formalisms rely on existing semantic models : E/R based and NIAM models are the most often used with some extensions (see [SORE88], [WELK92b], [SMOL91], [HOFS93]). Harmsen and Saeki [Saek96] have compared four languages, namely Object-Z, MEL, GOPPRR and HFSP. The selection of the four languages was justified by the fact that they are representative of four schools of thought. GOPRR belongs to the data oriented school stressing the representation of the product aspect of methods. Object-Z takes a similar position but emphasises object oriented modelling. The third school consists of languages developed for software process modelling and capturing design rationale. HFSP belongs to this group. Finally MEL is a specific method engineering language.The conclusion of the study is not clear and the claim is that ME languages should themselves be obtained by assembly of various languages fragments in order to be purpose-fit.

We certainly need more experiences in representing methods to define the set of requirements for a "good" ME language. It is clear also that current attempts have focussed on the representation of the product aspect of methods and have been using languages dealing with data and data constraints representations. The external facet of the language has to be distinguished from its underlying computational model. Thus, regarding the latter, it is not clear why a first-order predicate logic or a temporal logic should not suffice to represent the semantics of methods representations. Regarding the former, perhaps more emphasis on graphical notations is required.

Method components

One proposal for method components [HARM94] viewed a method as a collection of method fragments. A fragment can be either a *product fragment* or a *process fragment*. A product fragment captures product related knowledge of methods whereas a process fragment captures activity related knowledge. Product fragments are deliverables, documents, models, diagrams or concepts. The concept of class or of a binary relationship between two entities, or the entire E/R model are examples of product fragments. Process fragments are stages, activities and tasks to be carried out. Guidelines to "Create an E/R schema' or to 'Perform Requirements engineering" are examples of process fragments.

The drawback of the fragment based approach is the over-emphasis on the product fragment resulting in underdeveloped meta-process modeling. In addition, the process models underpining the meta-models are often activity-based [WIJE91], [MART94].

[ROLL94b], [ROLL95], [ROLL94a], [PLIH95] within the ESPRIT project NATURE, proposed a decision-oriented process meta-model which places equal emphasis on the product and process aspects of methods. A theoretical formulation is available in [SCHW95]. The proposal defines a method component as a tight coupling between these aspects. This is achieved by organising components as *chunks* defined around the notion of context (figure 4). A *Context* is a couple <situation, decision>, where the *decision* part represents the choice an IS developer can make at a moment in the engineering process and the *situation* is defined as the part of the product it makes sense to make a decision on. A decision corresponds to an intention, a goal that the developer wants to fulfil. The chunk prescribes the way to proceed in this situation to make the decision. There are three *ways* of proceeding: to execute an action which transforms the product, to make a choice among a set of predefined possibilities and to perform a plan for decision making. These three correspond to the three types of contexts shown in Figure 4, namely, executable, choice and plan contexts.

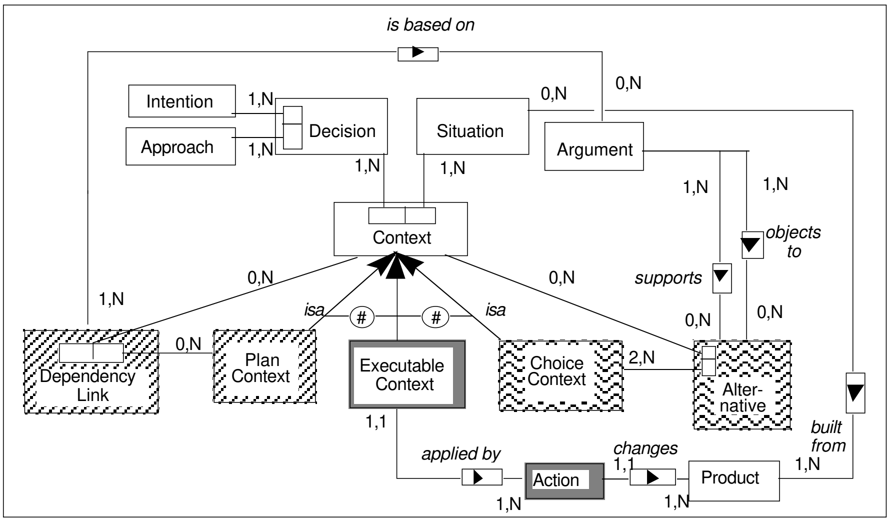

Figure 4 : Overview of the NATURE process chunk

To our knowledge, these are the only two proposals for defining method components. It is clear that additional work is needed before an agreed notion of method components can be arrived at.

Method components should be looked upon according to *two other perspectives* : *abstraction* and *granularity*. Method components are expressed with different granularity, at various levels of abstraction. For example consider the following method components (1) the OMT methodology, (2) the ER modelling approach, (3) the rules to define the key of an Entity-Type, (4) a generic outline providing a stepwise organisation of system analysis and (5) a generic set of guidelines for any concept description. The granularity is larger in (1) and (2) than in (3) above. Further, the first three are expressed at a less abstract level than the fourth and fifth examples above.

Notice that depending on its level of abstraction the method component will be reused as such (perhaps after some customisation) or will be instantiated before being assembled in the method under construction. Components of examples (1) to (3) above are directly reusable whereas each generic activity of the outline (example (4)) has to be instantiated according to the specific product of the method in hand before being used. Similarly the generic guidelines for concept description (example (5)) requires instantiation for each particular concept of the method under construction.

[VANS96] proposes two kinds of method components at two different levels of granularity, called *route map* and *method fragments* respectively. A *route map fragment* is "a complete route map of a system development method". It refers to strategies such as delivery, development, realization etc strategies, activities and products concerning system develoment as well as project management. The *method fragment* is "a coherent part of a method for system development or project management". Method fragments may be linked to a route map which may establish a complete project approch or a situated method.

[ROLL96b] introduce the notion of *framework* to model commonalties among methods and the notion of a *method construction pattern, pattern* for short, to capture generic laws governing the construction of different but similar methods. A framework is a method chunk which formalizes, in a more abstract way than a component does, knowledge which is common among several methods. A pattern models a common behaviour in method construction. It is generic in the sense that it is used by a typical method engineer in every method construction process. It is more abstract than a component or a framework. Both terms have been chosen by analogy to reuse approaches in the object oriented area. Patterns are there defined as solutions to generic problems which arise in many applications [GAMM93], [PREE95] whereas a framework is application domain dependent [WIRF90], [JOHN88].

Different kinds of construction patterns have been defined, like the *Identify, Describe*, *Construct*, *Define, Check and Refine patterns* [ROLL96a]. These patterns can be regarded as generic laws underlying method construction; these laws are generic in the sense that they can be applied to the construction of many methods. Using these laws, six of the traditional analysis methodologies, OMT, OOA, SA/SD, ER, O* and OOD have been generated.

Perhaps what is needed is a corpus of both, generic *method knowledge* and generic *method construction knowledge* which, has as yet, not been looked for, identified and described. A suggested research topic is therefore to develop a *domain analysis* approach to identify objects, rules and constraints which are :

(a) common among different (but similar) methods

(b) common among different (but similar) ways of method construction

and to formalize them as method chunks. In this way, method engineering can use the results of method domain analysis and save a significant amount of time as demonstrated in other domains [ARAN89]. Assuming that the degree of similarity which exists in the construction of methods which belong to the same area is similar to the equivalent degree in system requirements engineering, then method domain analysis can result in significant overall productivity improvement in method construction. Indeed, Jones [JONE84] indicates that only 15% of the requirements for a new system are unique to the system; the remaining 85% comes from the requirements of existing similar ones.

Component characterisation

Assuming that method components exist in a method base, the question now is "how to deliver the relevant method components to the user?" The ME community has been looking at this question in two ways : first, by promoting a global analysis of the project on hand based on contingency criteria and, secondly, by associating descriptors to components in order to ease the retrieval of components meeting the requirements of the user. Therefore in the former the project situation is at a very global level whereas in the latter the descriptors of method components support local matching with the situation at hand. Besides it is clear that the issue of component charaterisation is in a preliminary state of resolution.

As an example of the first approach let us consider the contingency model of [VANS96]. The model is based on 17 contingency factors which take value between *Low* and *High*. These are :

- Management commitment (for the IS project)
- Importance (of the project)
- Impact(of the project)
- Resistance and conflict (to what extent stakeholders have different or conflicting interests)
- Time pressure
- Shortage of human resources
- Shortage of means
- Formality (of the project procedures)
- Knowledge and experience(of the project team)
- Skills
- Size
- Relationships (between the IS under development and existing IS)
- Dependency (of the project to external factors)
- Clarity (of the project goals, obectives etc)
- Stability (of the project goals)
- Level of innovation

According to the authors, the characterisation of the project allows them to select the method components appropriate for the project. The approach has been tried out in nine non-standard projects of the systems development departement of a bank organisation.

The second approach [ROLL96b] uses the notion of *descriptor* [DeAnt91] as a means to describe method chunks. A descriptor plays for a method chunk the same role as a meta-class does for a class. It is similar to the one of faceted classification schema [PRIE87] developed in the context of software reuse. As method chunks, descriptors are organised in a contextual fashion : each of them categorizes the situation in which the chunk can be used and describes the intention of its use. A descriptor can be seen as a *meta-context* which links the situation in which a method chunk is relevant to the intention the chunk allows to fulfil. The situation part refers to the characteristics of the projects in the development of which the chunk can be used as part of the project method. The intention part refers to the engineering intention(s) that could be fulfilled when using the chunk. As shown in figure 5, a method chunk is said to be relevant *for* (relationship-type *for* in Figure 5) a certain *situation* (entity-type *situation*) *to* (relationship-type *to*) achieve a certain *intention* (entity-type *intention*). For example, Chunk c is applicable *for* *high risk project* (characterization of the situation) *to* *a reengineering purpose* (intention).

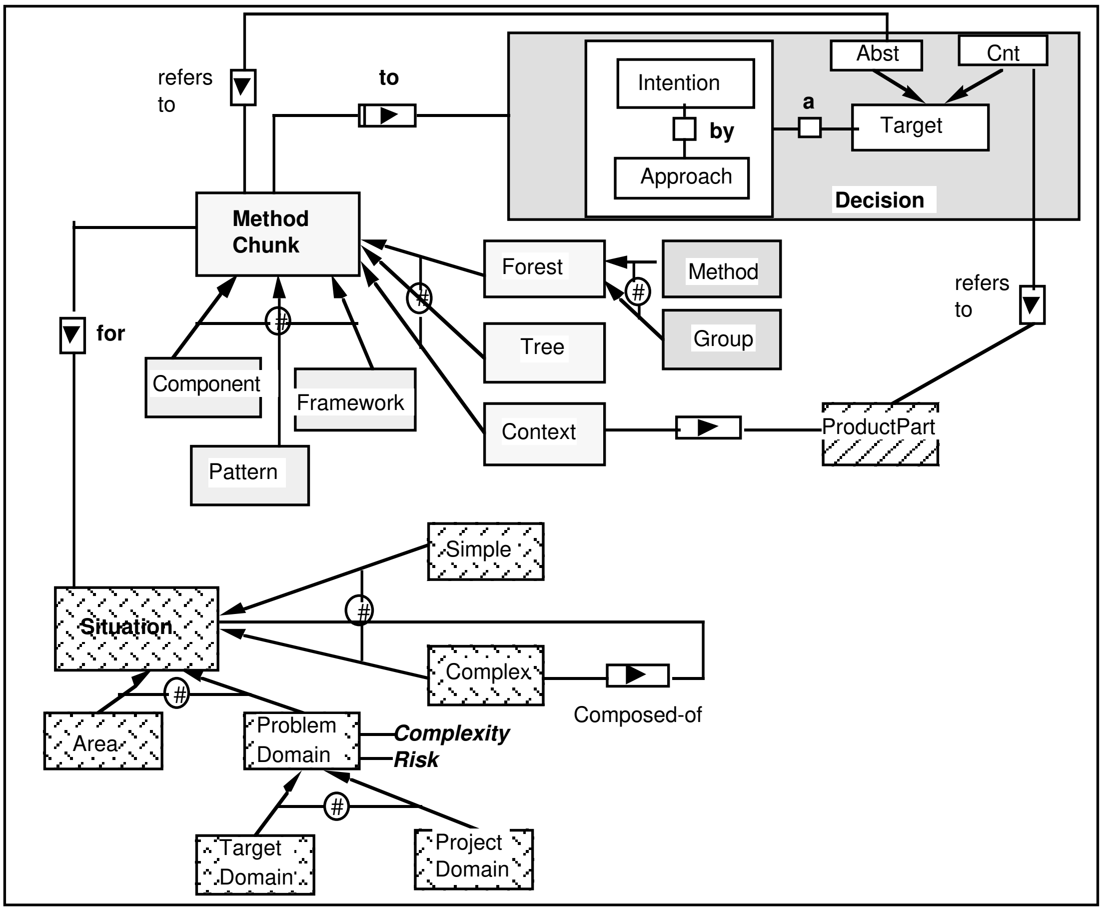

Figure 5 Descriptor of a method chunk

Based on the notions of *Intention* ,*Approach* ,*Target* and*Product Parts.* of the NATURE contextual modelling approach [ROLL95], the intentional part of the descriptor can express that a chunk is relevant *to* (relationship-type *to* in Figure 5) achieve *a* (relationship-type *a* in Figure 5) targeted intention *by* a given approach. Therefore the previous example can be refined in the following way : Chunk c is applicable *for* *high risk project* *to* *reengineer* (intention) *a* *business process* (target) *by* *domain based approach* (approach).

The situation part of a descriptor aims at providing the means to evaluate the adequacy of the method chunk to the situation of project at hand while the intention part tries to ensure that the goal of the project matches the goal of the method chunk. As shown in Figure 5 the situation is characterised in two ways : (a) by the *Area* (entity-type *Area*) of the project and, (b) by the *complexity* and *risk* (properties of the entity-type *Problem Domain)* as two situational factors evaluated for both the *Project Domain* (entity-type *Project Domain*) and the *Target Domain* (entity-type

Target Domain). The characterisation of the Problem Domain is based on the results achieved by the EUROMETHOD project [FRAN94].

The research related to the system world has so far been focussing on meta-modelling issues. This is understandable as it forms the basis for systemic ME. However meta-product modelling has been favoured at the expense of meta-process modelling. In addition the way meta-models are built is essentially a direct modelling of what exists today in methods. As we will argue in the next section an effort should be made to abstract from existing methods their common features more than just modelling those features. Finally, research on the criteria or factors which characterise the method situation is in a very early stage of development.

5. Usage world

The usage world deals with with the *intentionality of method engineering*. IS methods are generally assumed to be situation-independent. However, there are a multitude of different development methods each having their own advantages and disadvantages relating to the problem domain or the development context. In addition, experience with method use shows that system developers adapt and modify the methods they use to the situation and to their own personal preferences. Developers may need to create a new method from scratch, modify (e.g incrementally improve or tailor) an existing method, or reuse parts of various methods and recombine them to create the new, preferred method.

The goal of method engineering is not only to "operationalize" methods but also to correct general oversights in many of the current published methods. Method engineering introduces a third level of abstraction in information system development, a method for creating methods (figure 6). The challenge of ME is to provide means to increase our understanding of the general factors which must be adapted to the development situation. Additional work is needed to determine what factors should be included at the third level. Examples might include usability, availability, security, etc. Clearly this requires not only studying methods (which is maybe the focus of ME nowadays) but studying the reality of information systems development. In other words before building meta-models which reflect old models on a new level of abstraction, one should question the old ones. The benefit of ME is dependent of our ability to understand why the third level should resolve problems which were not solved by the two other levels.

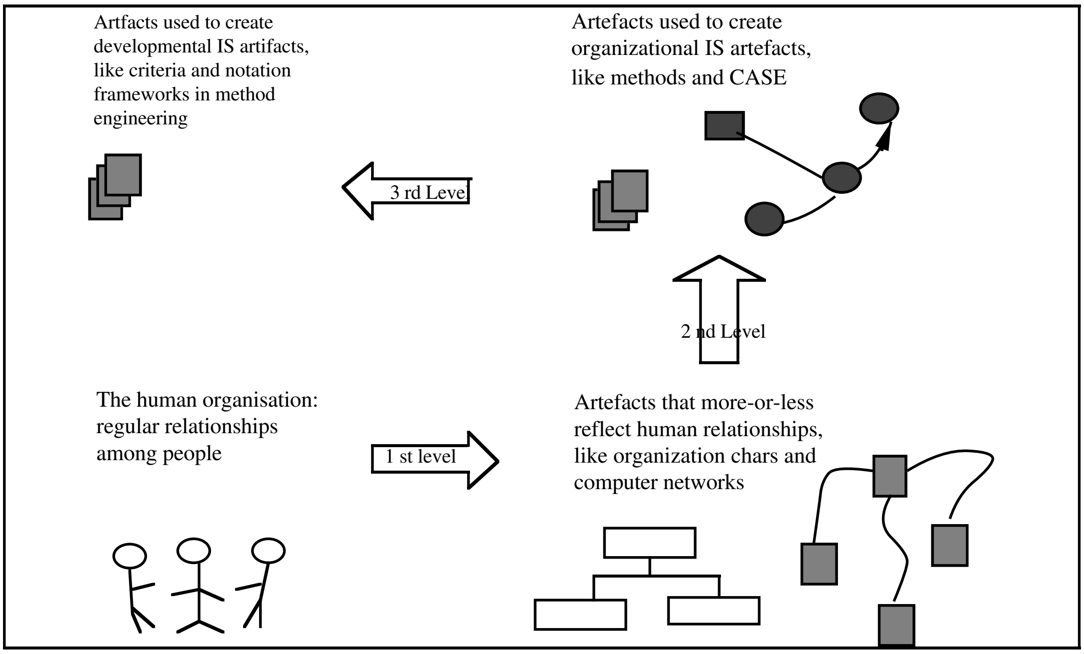

figure 6 The three ISD levels (adapted from Baskerville)

For instance it is a fact that IT is ineffective in various ways, outdated, misplaced or not used as planned because it is rejected by the organisation users (see literature on failures e.g Lyytinen 1988). The IS community introduced IS methods (the second level) to avoid the failures of the first level.But one also discovers that IT development *methods* can be misused, or unused, misplaced and outdated. ME raises the problems at a higher level. The challenge is first, to understand the general factors which influence the second and third levels. Secondly, the third level must provide general processes which based on these factors are able to generate a method and consequently a system adapted to the project at hand.

Another argument developed by [BASK96] is that the third level of abstraction introduced by ME (figure 6) can help in understanding the relationships between human organisations, their organisational structure and structural artifacts. Structural artifacts include IT, ISD methods and method engineering. The claim is that there are conflicts between such artifacts and the organisation and between the artifacts themselves that can be resolved or, at least, better understood through the introduction of the third level. The previously mentioned failures or inadequacies of ISs to organisations are examples of such conflicts.

6. Development world

The development world is concerned with *the process of constructing methods.* If we draw an analogy with tools for advanced software process management (e. g. software process centred environments) then the method engineering process should follow the framework [DOWS93] illustrated in figure 7. The framework has three different sub-domains : a process model domain, a process performance domain; and a process model enactment domain which interact with each others.

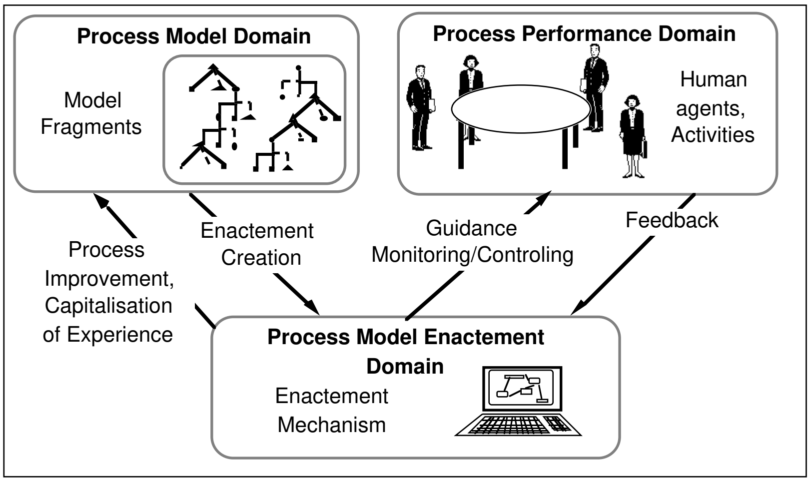

Figure 7: process domains

The *process model domain* contains process models or parts of process models.

The *process performance domain* encompasses the actual activities conducted by human agents and machines as well, in the course of a project. Some will be executed by software tools; others will consist of thinking, writing, exchanging ideas, taking decisions through formal and informal interactions between members of the project team. All these activities are sought to be supported by the process model.

The *process model enactment domain* is concerned with what takes place in the CASE environment to support process performance governed by the process model. It includes run-time occurrences of process models or parts of process models which are called enactments.The enactment mechanism uses the process model to determine and control the interactions with the agents performing the process and with other components of the CASE environment, so as to support, guide, or enforce performance of the process in a way consistent with the process model.

The three domains interact with each other. Firstly, the process model influences the way in which the process is performed. Actual performance will then correspond, to some extent, to the model of how it should be performed. Secondly, the course of enactment may need to be contingent on events arising from actual process performance. Therefore the process will be different from the theoretical instantiation of the process model. This leads to the idea of feedback from process trace to process model.allowing its improvement.

Current work in method engineering has not yet reached this level of sophistication. One option for situational method engineering [HARM94] is depicted in figure 8 as a data flow diagram.The process starts with the characterisation of the project environment which includes the existing systems development organization, the customer organization, the supplier organization, the area of application, information and computerizaton policies, etc. Contextual and contingency factors derived from the project environment are important for supporting the selection of the appropriate method components from the method base repository.

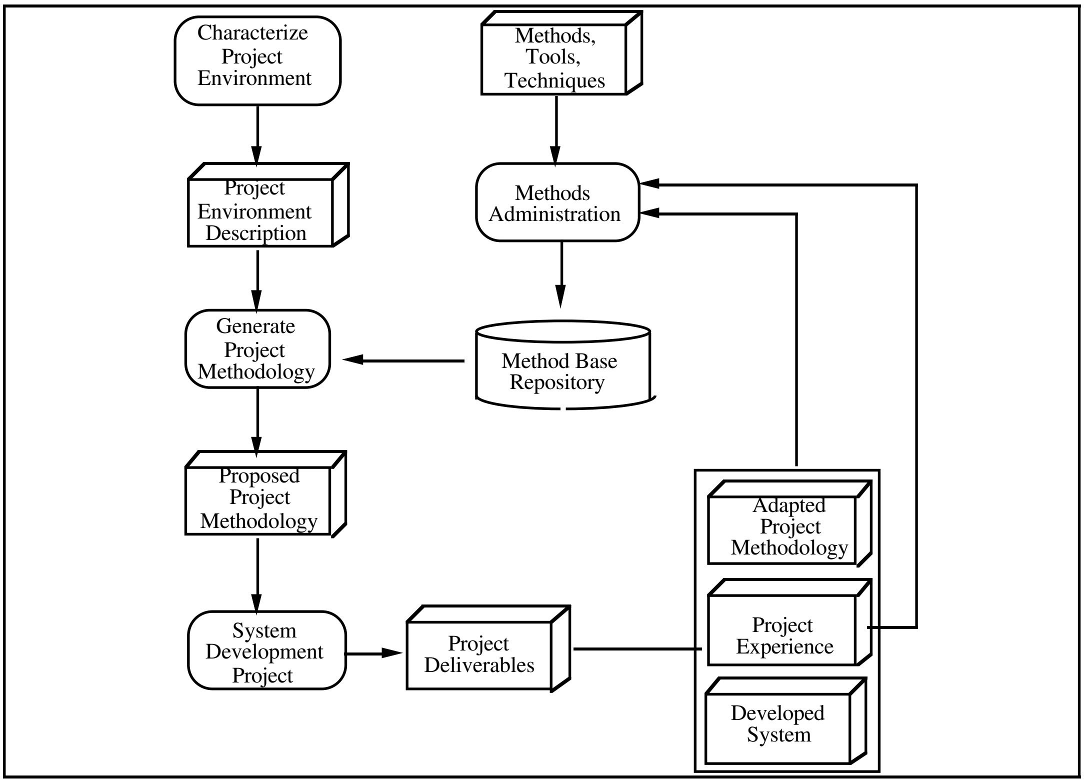

figure 8: A data flow diagram specifying the process of situational method construction

The contingency factors are determined during the project characterisation as a result of interviews, brain storming sessions, questionaires or other knowledge acquisition techniques.The important contingency factors are utilized to select the appropriate components from the method base repository. The situated method is constructed based on these fragments. The construction is supported by rules to assemble components and constraints to be satisfied by the constructed method to be consistent. Rules and constraints are parts of the knowledge stored in the method repository to support the situated method construction process. Then, an information system project is started, using the situated method constructed in the previous phase and the deliverables of the project are produced. Evaluations during and after the project performance may yield new knowledge about situated method development, which is captured in the method base repository. Finally new method components can be added to the repository for further use.

The method construction process calls for software support. Computer Aided Method Engineering (CAME) tools have been designed and developed for this purpose. Computer Aided System Engineering (CASE) tools support the development of information systems. CAME tools aim at supporting the development of methods. Using [HARM94] as a basis, the suggested functionality of a CAME tool is as follows:

- definition and evaluation of contingency rules and factors or descriptors. As discussed earlier, this enables the right choice of the method components

- storage of method components, method construction knowledge, past experience, heuristics etc in a repository called the method base.

- retrieval of the contents of the method base. A query language for accessing the contents of the method base needs to be defined.

- composition of method components. The knowledge permitting the development of a new method must be available.

- validation and verification of the constructed method. The CAME tool should not only support the selection and assembly tasks but also check the resulting method. The tool, therefore, should incorporate guidelines to ensure the correctness of the method.

- adaptation facilities for modification of the contents of the method base as a result of the experience gained

- support and guidance of the method engineering task.

It is possible to establish a relationship through the repository between the method and application engineering environments. This is shown in Figure 9. The repository extends the one advocated in Information Resource Dictionary Framework Standard [IRDS90]. Similar to the IRDS repository, it consists of three levels. However, whereas the IRDS deals with levels of *product* description, the repository deals with levels of both, *product and process* descriptions .

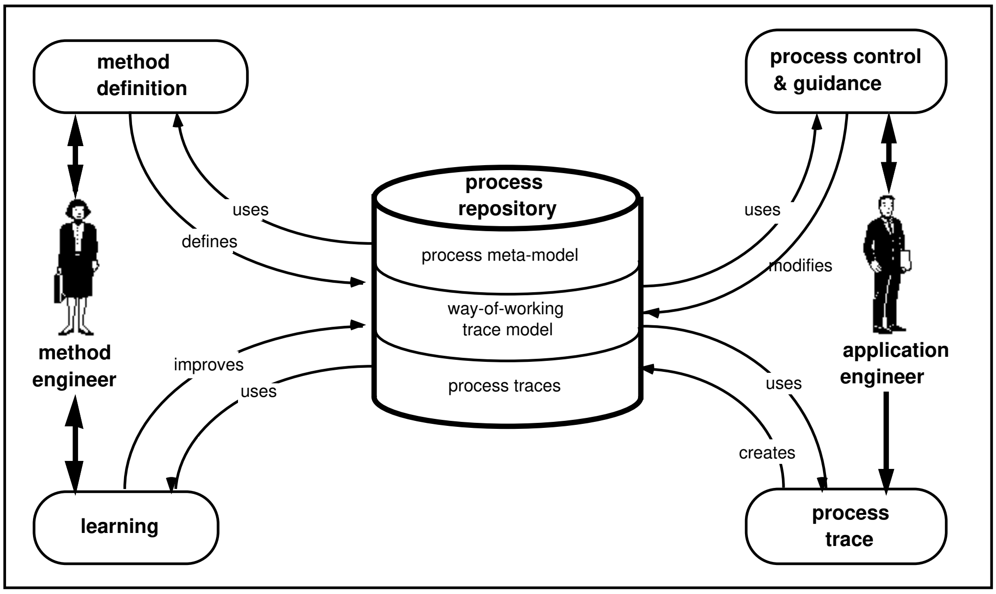

Figure 9 : Repository based and process-integrated environment support

The environment is composed of two sub-environments, the *application engineering environment* in which the process is guided, executed, and traced, and the *method-engineering environment* in which the process is defined and improved. These two environments use the *process repository* which contains the information necessary to provide the intended functionality.

This architecture assumes the existence of an enactment mechanism which supports both, the enactment of the application development and of the method of development. It, therefore, assumes that the method development process has been completely modelled in order to be executed or interpreted by the enactment mechanism. It emphasises the role of tracing to support the improvement of the process models by utilising previous experience.

While there is now consensus on the functionality that a CAME tool should provide, considerable work has still to be done to achieve implementations meeting this functionality. However, a number of meta-CASE products and prototypes have been developed which implement this functionality partially. As examples consider Maestro II [MERB91], MetaEdit+ [KELL96], Decamerone [HARM95], and Mentor [SISA96]. Decamerone is in its early stages of development but aims at supporting the assembly of product and process fragments which have been selected based on project contingency factors. However, it does not provide guidance either in method engineering or in application engineering. MetaEdit+ includes a number of instantiations of mainly the product aspects of about twenty methods. Maestro II is a meta-CASE tool on which Decamerone has been developed. The emphasis in Mentor has been on the unification of the process and product aspects of methods as well as on the guidance and support that can be provided in both the method engineering and the applciation engineering work.

None of the tools referred to above is really tackling the problem of assembly of components in a consistent and coherent way. Similarly, the retrieval of components is not well-supported. The contents of the method base are highly restricted and need to be extended to include, for example, knowledge about the situated usability of components. The need for tools to support tracing of the method engineering process is critical to the ability to subsequently learn and improve this process. However, little attention has been paid to this aspect. The ability to dynamically change the method has also not been explored.

To summarise, the design of Computer Aided Method Engineering (CAME) environments is a research issue involving a number of different problems such as *repository structuring* and *management*, *enactment mechanisms*, efficient *interpretation/execution* of process modelling languages, process descriptions *configuration management.*

7. Conclusion

This paper has extended the four worlds framework to understand the subject of methods, their representations, the development of computer based support for the construction of methods, as well as the rationale for the discipline of method engineering.

We have seen a plethora of methods in the subject world. However, there is a felt need for empirical evaluation of the use of methods. Besides, current studies show that there is a need to situate methods, i.e, define methods as context dependent whereas they are today considered to be domain independent.

Representation of methods is based on meta-modelling around which the whole area of method engineering has developed. The more modern meta-models look for an integration of the process and product aspects of methods whereas earlier meta-models focussed on product aspects only. Meta-modelling per se does not tackle the important problem of modular description of methods. Therefore, there is a need for further research to define the notion of a method component better.

In the development world we have highlighted the need for refinement of the existing architecture of CAME environments and their integration with CASE environments. Besides, the need to integrate in both, CAME and CASE, enactment mechanisms to support process execution was shown. Finally even though the functionality of CAME tools has been rather well identified, implementation of tools with this full functionality has yet to be achieved.

In the usage world the question of why we should use a meta approach in method definition has ben addressed. It was pointed out that the straight-forward modelling of current methods is inadequate for solving any of the unsolved problems of IT acceptance in an organisation.

The challenge of method engineering is to understand why these problems are unsolved, relate them to organisational factors, and adapt methods to develop IT systems to the specific factors of the situation at hand.

8. References

[AAEN92] Aaen et Al, A tale of two countries:CASE experience and expectations, The Impact of Computer Supported Technology on Information Systems Development, North Holland Pub, pp 61-93, 1992

[AKHR92] F. N. Akhras and S.S.S. Melnikoff, Towards Dynamic generation of Knowledge-Based Environments for Software Process Assistance, 1992

[ARAN89] Arango, Domain analysis : from art to engineering discipline, Proc. 5th Int. Workshop on Software Specification and Design, IEEE Computer Society Press, San Diego, 1989

[ARME93] P. Armenise, S. Bandinelli, C. Ghezzi, A. Morzenti, A survey and assessment of software process representation formalisms, Int. Journal of Software Engineering and Knowledge Engineering,Vol. 3, No. 3, 1993

[BASK96] Baskerville R, Structural artifacts in method engineering: The security imperative, IFIPWG 8.1 Conference on Method Engineering, Chapman and Hall, pp 8-28, 1996

[BENI56] Benington H.D.; Production of Large Computer Programs, Proc. ONR Symposium on Advanced Programming Methods for digital Computers, pp 15-27, Juin 1956

[BOEH88] B. Boehm, A Spiral Model of Software Development and Enhancement, IEEE Computer Vol. 21, No. 5, 1988

[BRIN90] S. Brinkkemper, Formalisation of Information Systems Modelling, Ph.D. thesis, University of Nijmegen, Thesis Publishers, Amsterdam, 1990

[BUBE94] J. Bubenko, C. Rolland, P. Loucopoulos, V. DeAntonellis Facilitating "Fuzzy to Formal" Requirements Modelling, IEEE 1st Int. Conference on Requirements Engineering, ICRE'94, pp 154-158, 94

[CAMP94] Campbell L. Halpin T., Abstraction techniques for conceptual schema, in Proc. 5th Australasian Database Conference, Global Publications Services, New Zealand, p 374-388, 1994

[DARD91] Dardenne, A., Fickas, S., van Lamsweerde, A., Goal-directed concept acquisition in requirements elicitation, Proc. 6th IEEE Workshop System Specification and Design0 , Como, Italy, 14-21, 1991

[DEAN91] De Antonellis V., Pernici B., Samarati P. (1991) "F-ORM METHOD : A methodology for reusing specifications", in Object Oriented Approach in Information Systems, Van Assche F., Moulin B., Rolland C. (eds), North Holland, 1991

[DOWS88] M. Dowson, Iteration in the Software Process, Proc 9th Int. Conf. on "Software Engineering", 1988

[DOWS93] M. Dowson, Software Process Themes and Issues, IEEE int. conf. on the Software Process, Berlin, 1993

[FALK94] Falkenberg E.D., Oie G.L.H., Meta-model hierarchies from an object-role modelling perspective, in 1st Intl. Conference on Object-Role Modelling, Univ. of Queensland, Brisbane, Australia, pp 310-323, 1994

[FINK90] Finkelstein A., Kramer J., Goedicke M., ViewPoint Oriented Software Development, Proc. Conf. "Le Génie Logiciel et ses Applications", Toulouse, p 337-351, 1990.

[FINK94] A. Finkelstein, J. Kramer, B. Nuseibeh (eds), "Software Process Modelling and Technology", John Wiley (pub), 1994.

[FRAN91] M. Franckson, C. Peugeot, "Specification of the Object and Process Modeling Language ", ESF Report n° D122-OPML-1. 0, 1991

[FRAN94] Franckson M., The Euromethod deliverable model and its contribution to the objectves of Euromethod, Proc. IFIP-TC8 Int. Conf. on Methods and Tools for the Information Systems Life Cycle, Verrijn-Stuart and Olle (eds), North-Holland, pp131-149, 1994

[GAMM93] Gamma E., Helm R., Johnson R., Vlissides J., Design patterns : Abstraction and Reuse of Object-Oriented Design, Proc. of the ECOOP'93 Conf., Sringer Verlag, 1993

[GRUN96] Grundy J.C., Venable J.R., Towards an integrated environment for method engineering, Proc. IFIP WG 8.1 Conf. on method Engineering, Chapman and Hall, pp 45-62, 1996

[HARM94] Harmsen F et al, Situational method engineering for informational system project approaches, in Method and Associated Tools for the Information Systems Life Cycle, Verrijn-Stuart and Olle (eds.), North Holland, pp169-194, 1994

[HARM95] Harmsen F., Brinkkemper S., Design and implementation of a method base management system for situational CASE environment. Proc. 2nd APSEC Conference, IEEE Computer Society Press, pp 430-438, 1995

[HARM96] Harmsen F., Saeki M., Comparison of four method engineering languages, IFIP 8.1 Conference on Method Engineering, 1996, Chapman and Hall, pp 209-231, 1996

[HEND90] B. Henderson-Sellers, J. M. Edwards, The Object-oriented Systems Life-Cycle, Comm. of the ACM, 09, 1990

[HIDD94] Hidding G.J., Methodology information : who uses it and why not?, Proc. WITS-94, Vancouver, Canada, 1994

[HIRS92] Hirschheim R. and Klein H.K. Paradigmatic influences on information systems development methodogies:evolution and conceptual advances, Advances in Computers, 34, pp. 294-381, 1992

[HOFS93] HOFStede A.H.M., Proper H.A., van der Weide Th. P., Formal definition of a conceptual language for the description and manipulation of information models, Information Systems 18, pp 489-523

[HUMP89] Humphrey W.S., Kellner M.I., Software Process Modeling: Principles of Entity Process Models, Proc. 11th Int. Conf. on Software Engineering, 1989

[HONG93] Hong S.G. van der Goor, Brinkkemper S., A comparison of six object-oriented analysis and design methods, 26th Hawaiian Conference on System Sciences IEEE Computer Society Press, 1993

[IIVA90] J. Iivari, Hierarchical Spiral Model for Information System and Software Development, Information and Software Technology, Vol. 32, Part 1 : "Theoritical Background", No. 6, and Part 2 : "Theoritical Background", No. 7, 1990

[IIVA94] J. IIvari, Comparing O.O. Methods, Int. IFIP WG8.1 Conference in CRIS series : Method and associated Tools for the Information Systems Life Cycle", 1994.

[IRDS90] Information Technology - Information Resource Dictionary System (IRDS) - Framework, ISO/IEC International Standard, 1990

[JACK84] Jackson M.C., Keys P., Towards a system of systems methodologies, Journal of the Operational Research Society, 35, pp. 473-486, 1984

[JARK92] M. Jarke et al DAIDA - An Environment for Evolving Information Systems; ACM Trans. on Information Systems, Vol. 10, No. 1, 1992.

[JARK93] Jarke, M., Pohl, K., "Establishing visions in context: towards a model of requirements processes". Proc. 12th Intl. Conf. Information Systems, Orlando, Fl, 1993.

[JARK94] M. Jarke, K. Pohl, C. Rolland, J.R. Schmitt, Experienced-Based Method Evaluation and Improvement : A Process Modeling Approach, Int. IFIP WG8.1 Conference in CRIS series : Method and associated Tools for the Information Systems Life Cycle", 1994.

[JONE84] Jones T.C., Reusability in programming : a survey of the state of the art, IEEE Transactions on Software Engineering,SE Vol 10, No1, 1984

[JOHN88] Johnson R. E., Foote B., Designing reusable classes, Journal of Object-Oriented Programming, Vol 1, No3, 1988

[JOHN91] Johnson R.E., Russo F., "Reusing object-oriented design", Technical report UIUCDCS 91-1696, University of Illinois, May 1991

[KELL96] Kelly S., Lyyttinen K., Rossi M., Meta Edit+: A fully configurable, multi-user and multi-tool CASE and CAME environment, Proc. CAiSE'96 Conference, Springer Verlag, 1996

[KOTT84] Kottemann J.E., Kosynski B.R., Dynamic meta-systems for information systems development, Proc. 5th Intl. Conference on Information Systems, pp 187-204, 1984

[LUBA93] Lubars et al, M. Lubars , C. Potts, C. Richter, A Review of the State of the Practice in Requirements Modeling, Proc. Int. Symposium on Requirements Engineering, 1993.

[LYYT87].Lyytinen K., Different perspectives on information systems : problems and solutions, ACM Computing Surveys, Vol 19, No1, 1987

[LYYT89] Lyytineen et al, K. Lyytinen, K. Smolander, V-P. Tahvainen, Modelling CASE Environments in sytems Work, CASE'89 conference papers, Kista, Sweden, 1989.

[LYYT97] Lyytinen K. Hirschhiem A., Information systems failures: A survey and classification of the empirical literature, Oxford Survey in Information Technology 4, 1997

[MART94] P. Marttiin, Methodology Engineering in CASE shells : Design Issue and current Practice, PhD thesis, Computer science and information systems reports, Technical report TR-4, 1994.

[MART95] Martiin P. et al, Modelling requirements for future CASE:issues nd implementation considerations, Information Resources Management Journal, 8, 1, pp 15-25, 1995

[MERB91] Merbeth G., Maestro II- das intergrierte CASE-system von Softlab, CASE systeme and Werkzeuge (Ed. H. Balzert) BI Wissenschaftsverlag, pp 319-336, 1991

[MORE93] Moreno, M, Souveyet, C., The Evolutionary Object Model (EOM), IFIP WG 8.1 Conf. on Information Systems Development Process, Como, Italy, Sept. 1993

[NADI87] Nadin M., Novak M.; “MIND: A Design Machine, Conceptual Framework”; Intelligent CAD Systems I, Springer Verlag, 1987

[OIE92] Oie G.L.H van Hemmen, Falkenberg E.D., Brinkkemper S., The meta-model hierarchy:a framework for information system concepts and techniques, University of Nijmegen, 1992

[OLLE88] Olle T. W., J. Hagelstein, I. MacDonald, C. Rolland, F. Van Assche, A. A. Verrijn-Stuart, Information Systems Methodologies : A Framework for Understanding, Addison Wesley, 1988

[OSTE87] L. Osterweil, Software processes are software too; Proc. 9th Int. Conference on Software Engineering, IEEE Computer Society, Washington, DC, 1987, pp2-13, 1987

[PLIH95] Plihon V., Rolland C., Modelling Ways-of-Working, Proc 7th Int. Conf. on Advanced Information Systems Engineering, CAISE'95, Springer Verlag, 1995

[POTT89] Potts C., A Generic Model for Representing Design Methods, Proc. 11th Int. Conf. on Software Engineering, 1989

[POTT94] Potts C. and Al, Inquiry-based Requirements Analysis, IEEE Software, Mars 1994

[PRAK94] Prakash N., A Process View of Methodologies, 6th Int. Conf. on Advanced Information Systems Engineering, CAISE'94, Springer Verlag, 1994

[PREE95] Pree W., Design Patterns for Object-Oriented Software Development, Addison Wesley, 1995

[PRIE87] Prieto-Diaz R., Freeman P., Classifying software for reusability, IEEE Software, Vol. 4, No. 1, 1987

[ROLL91] Rolland C., Cauvet C., ALECSI : An Expert System for Requirements Engineering, Proc. 3th Int. Conf. on Advanced Information Systems Engineering (CAISE'91), Springer Verlag, 1991.

[ROLL93] Rolland C., Modeling the Requirements Engineering Process, Information Modelling and Knowledge Bases, IOS Press, 1993

[ROLL94a] Rolland C., Modeling the evolution of artifacts, 1st IEEE Int. Conference on Requirements Engineering, Colorado Springs, Colorado, 1994.

[ROLL94b] Rolland C. and Prakash N., A Contextual Approach to modeling the Requirements Engineering Process, SEKE'94, 6th International Conference on Software Engineering and Knowledge Engineering, Vilnius, Lithuania, 1994

[ROLL95] Rolland C., Souveyet C., Moreno M., An Approach for Defining Ways-Of-Working, Information Systems, Vol 20, No4, pp337-359, 1995

[ROLL96a] Rolland C., Plihon V., Using generic chunks to generate process models fragments in Proc.of 2nd IEEE Int. Conf. on Requirements Engineering", ICRE'96, Colorado Spring, 1996

[ROLL96b] Rolland C. and Prakash N., A proposal for context-specific method engineering, IFIP WG 8.1 Conference on Method Engineering, Chapman and Hall, pp 191-208, 1996

[ROLL97a] Rolland C., Numero special sur "L!ingénierie des procesus", ISI, 1997

[ROLL97] Rolland and Al, Deliverable D1-I1.1 CREWS, A proposal for a Scenario Classification Framework, 1997

[ROYC70] Royce W. W., Managing the Development of Large Software Systems, Proc. IEEE WESCON 08, 1970

[RUSS95] Russo et al, The use and adaptation of system development methodologies, Proc. 1995 Intl. Resources Management. Association Conference, Atlanta, 1995

[SAEK91] Saeki M., Kaneko T. Sakamoto M., A method for software process modelling and description using LOTOS, Proc. 1st Intl. Conference on the Software Process, IEEE Computer Society Press, Los Alamitos, CA, USA, pp 90-104, 1991

[SCHW95] Schwer S., Rolland C., Theoretical formalization of the process meta-modelling approach, internal CRI report 95-08, University of Paris 1, France, 1995

[SELI89] Seligman et al Seligmann P.S., Wijers G. M., Sol H.G., Analyzing the structure of I.S. methodologies, an alternative approach, in Proc. of the 1st Dutch Conference on Information Systems, Amersfoort, The Netherlands, 1989

[SISA96] Si-Said S., Rolland C., Grosz G., MENTOR : A Computer Aided Requirements Engineering Environment, in Proc 8th Int. Conf. on Advanced Information Systems Engineering (CAISE'96), Springer Verlag, 1996 .

[SMOL91] Smolander K., Lyttinen K., Tahvanainen V.P., Martiin P., MetaEdit: A flexible graphical environment for methodology modelling, Proc. 3rd Intl. Conference on Advanced Information Systems Engineering, Springer Verlag, pp 168-193, 1991

[SORE88] Soreson P.G., Tremblay J.P., McAllister A.J., The meta-view system for many specification environments, IEEE Software, March, pp 32-38, 1988

[TOMI89] Tomiyama T., Kiriyama T., Takeda H., Xue D., Yoshikaya H., Metamodel: A Key to Intelligent CAD Systems, Research in Engineering Design Vol 1, pp 19-34, 1989

[VANS96] Van Slooten K., Hodes B., Characterising IS development project, IFIP WG 8.1 Conference on Method Engineering, Chapman and Hall, pp 29-44, 1996

[VENA93] Venable G.R., CoCoA: a conceptual data modelling approach for complex problem domains, Ph.D. dissertation, SUNY, Binghampton, 1993

[WELK92a] Welke R.J, and Kumar K., Method engineering : a proposal for situation-specific methodology construction, in Systems Analysis and Design : A Research Agenda, Cotterman and Senn(eds), Wiley, pp257-268, 1992

[WELK92b] Welke R.J., The CASE repository: more than another data base application, in Challenges and Strategies for Research in Systems Development, Wiley, Chichester, U.K, 1992

[WIER91] Wieringa R.J., Combining static and dynamic modelling methods, Proc. IFIP WG 8.1 Conference on Objected Oriented Information Systems, North Holland, 1991

[WIJE90] Wijers G. M., van Dort H.E., Experiences with the use of CASE tools in the Netherlands, Advanced Information Systems Engineering, pp 5-20, 1990

[WIJE91] Wijers G.M., Modelling support in information systems development, Ph. D. thesis, Delft University of Technology, Thesis Publishers, Amsterdam, 1991

[WILS91] Wilson D.A, Rosenstein L.S., Shafer D., Programming with MacApp, Addison-Wesley, 1991

[WIRF90] Wirfs-Brock J., Johnson R., Surveying current research in Object-Oriented Design, Communications of ACM, Vol 33, No9, 1990

[WYNE93] Wynekoop J., Russo N., System development methodologies:unanswered questions and the research-practice gap, in Proc.14th Intl. Conf. Inf. Syst., New York, ACM Pub. pp 181- 190, 1993

[YOUR92] Yourdon E., The decline and fall of the american programmer, Prentice Hall, Englewood Cliffs, NJ, 1992

[YU94] Eric SK Yu, John Mylopoulos. From ER to AR_ modelling strategic Actor Relationships for Business Process Reengineering, In Proc. of the 13th International Conference on the Entity-Relationship Approach - ER'94, Manchester (UK), December 13-16, 1994.