CREWS Report Series 98 - 18

A COMPREHENSIVE VIEW OF PROCESS ENGINEERING

C. Rolland

University Paris-1 Sorbonne, 17, rue de la Sorbonne, 75005 Paris cedex 5,
FRANCE

email : rolland@univ-paris1.fr

Appeared in the proceedings of the 10th International Conference CAiSE'98, B. Lecture Notes in Computer Science 1413, Pernici, C. Thanos (Eds), Springer. Pisa, Italy, June 1998

A Comprehensive View of Process Engineering

Colette Rolland
University Paris-1 Sorbonne, 17, rue de la Sorbonne, 75005 Paris cedex 5,
FRANCE
email : rolland@univ-paris1.fr

Abstract. The paper proposes a faceted framework to understand and classify issues in system development process engineering. The framework identifies four different but complementary view-points. Each view allows us to capture a particular aspect of process engineering. Inter-relationships between these aspects allow us to show the influence that one aspect has on another.
In order to study, understand and classify a particular aspect of process engineering in its diversity we associate a set of facets with each aspect.
The paper uses the framework to raise questions, problems and research issues in the field.

1. INTRODUCTION

Process engineering is considered today as a key issue by both the software engineering and information systems engineering communities. Recent interest in process engineering is part of the shift of focus from the product to the process view of systems development. Process engineering is a rather new research area. Consequently there is no consensus on, for example, what would be a good formalism to represent processes in, or, even, on what the final objectives of process engineering are [Arm93] . However, 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 [Dow93], [Arm93] and [Jar94]. Studies of software development practices [Lub93], however, demonstrate that we know very little about the development process. Thus, to realise the promise of systems development processes, there is a great need [Dow93] for *a conceptual process engineering framework.*

In this paper we consider process engineering from four different, but complementary, view-points. Each *view* allows us to capture a particular aspect of process engineering. *Inter-relationships* between these aspects allow us to show the influence that one aspect has on another.

In order to study, understand and classify a particular aspect of process engineering in its diversity we associate a set of *facets* with each aspect. For example, in the development view, where the concern is with the way in which process models are developed, it is possible to turn to (a) the facet called *construction approach* to understand how a process model can be constructed, (b) the *construction technique* facet to understand how it can be engineered, (c) the *change support* facet to see how flexible the process model is etc..

Facets have been proposed by [Pri87] for classifying reusable components. They have also been used by [Rol98] in requirements engineering for understanding and classifying scenario based approaches. When used in process engineering, a facet provides a means for classification. For example, the coverage facet of the system world (see section 5 below) helps in classifying process models according to the underlying paradigm used: activity-oriented, product-oriented, decision-oriented or contextual. Each facet is measured by a set of relevant *attributes*. For instance, the description facet is measured by two attributes, the form and the notation attributes. Attributes have *values* which are defined in a *domain*. A domain may be a predefined type (INTEGER, BOOLEAN ...), an enumerated type (ENUM {x, y, z}), or a structured type (SET or TUPLE).

We use the four worlds framework as a baseline and attach (a) a view of process engineering to each of its worlds and (b) a set of facets to each view. As a result, it is possible to identify and investigate four major view points of process engineering: *what* are processes, *how* are they *represented*, how can their *representation be developed* and *used*, and, finally, what does *process engineering achieve*.
The multi-facet, multi-view approach adopted here makes it possible to look at process engineering in a comprehensive manner:
- *facets* provide an in-depth description of each aspect of process engineering whereas aspects give a view of process engineering in all its diversity.
- *relationships between facets* help in understanding the implications of one view on another.

2. THE FOUR-WORLDS FRAMEWORK

The four worlds framework originally proposed for system engineering has proved its efficiency in enhancing the understanding of various engineering disciplines, information systems engineering [Jar92], requirements engineering [Jar93], and method engineering [Rol97]. It can also help in understanding the field of process engineering which consists of applying engineering approaches, techniques, and tools to the construction of process models.

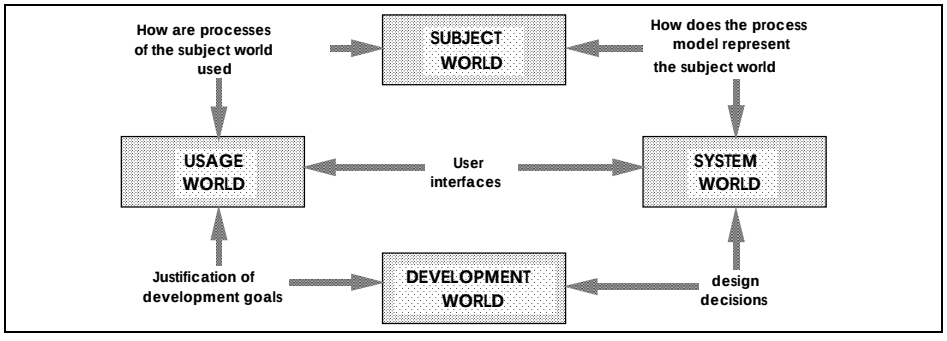

Fig. 1. The four worlds of process engineering

In the original system engineering framework (Fig. 1.), 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 as well as the mapping onto design specifications and implementations.
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 built whereas the subject world refers to the domain it shall represent.
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.

For our purposes, we identify the subject world as the world of processes. The *system world* deals with the *representation of processes* through process models. In the *usage world* we will investigate the reasons, the *rationale* for process engineering and relate the objectives of the users to the process models that can best meet these objectives. The *development world* deals with the *process of constructing process models*. This process is a meta-process in the sense that it supports the construction of processes, which in turn, will support the development of systems. The way the process might be supported by a tool environment is also a concern of this world.

The paper uses the four worlds to present the state of art in process engineering and to raise questions, problems and research issues in the field. It comprises four sections, each of these relating to one of the world. This allows us to discuss in a focused manner the different concerns of process engineering : the *definitions of processes*, 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

Our Universe of Discourse is the world of processes. In this world, it is of interest to look at the notion of a *process* and its *nature*.

A process is performed to produce a product. It has been described in the information systems area [Oll88] as the route to be followed to reach a product. This basic notion has been extended by [Poh93] who looks upon a product as a point in three-dimensional space comprising of the agreement, specification, and representation dimensions. Starting from some initial position in this space, the product moves through a succession of locations before a final position is reached. This final position corresponds to the desired product. The process then can be

considered to be the route starting from the initial product position and going through the succession of intermediate positions till the final product position is reached.

The term process has been defined differently in different coverage (see section V below for the notion of coverage). In the activity-oriented coverage it is defined as a related set of activities conducted for the specific purpose of product definition. In [Fei93] it is defined as "a set of partially ordered steps intended to reach a goal" and a process step is itself an atomic action of a process that has no externally visible sub-structure. In the product-oriented coverage, a process is a series of activities that cause successive product transformations to reach the desired product. [Fra91], [Hum89] and [Lon93] are three examples of definitions conforming to this view. In the decision-oriented coverage, a process is defined as a set of related decisions conducted for the specific purpose of product definition. This view has been developed, for instance in IBIS [Pot89], DAIDA[Jar92] and [Ros91]. Finally, in the coverage called context, a process is a sequence of contexts causing successive product transformations under the influence of a decision taken in a context [Jar93].

More intrinsically processes can be of different *kinds*. These various definitions reflect the multiple view points of the community about what is a process. However, these view points correspond to the various ways in which a process can be modelled and we will deal with in the system world.

Strategic processes are those that investigate alternative ways of doing a thing and eventually, produce a plan for doing it. There are many ways of doing the same thing and the best way, the one most suited to the situation at hand has to be found. Such processes are creative and alternative generation and selection from an alternative are very critical activities here.

Tactical processes are those which help in the achievement of a plan. As their name implies they are more concerned with the tactics to be adopted for actual plan achievement than with the development of a plan of achievement.

Implementation processes are the lowest level processes. They are directly concerned with the details of the *what* and *how* of plan implementation.

Thus, the subject world can be characterised by a facet having only one attribute called *Nature* defined as

Nature: ENUM{strategic, tactical, implementation}

As one can expect, we shall see below how the nature of the processes handled will influence the choice of a model adequate for their representation.

4. THE USAGE WORLD

The usage world is where the *goals of process use* are established and, consequently, the range of facilities required for process performance are determined. The usage world can be viewed [Dow93] as composed of three interacting domains : a *process model domain*, a *process performance domain*, and *a process model enactment domain* (Fig. 2.).

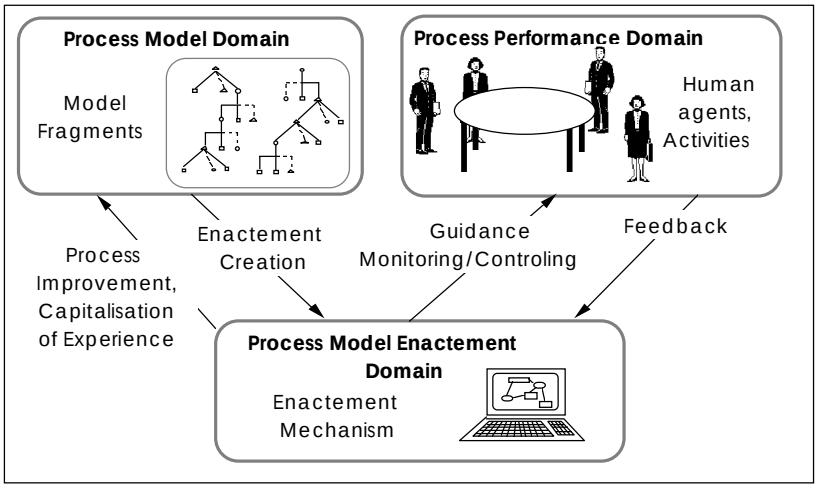

Fig. 2. Process domains

The process model domain contains process models. A process model describes the common properties of a class of processes having the same nature. The process performance domain deals with the actual activities conducted by human agents and machines, in the course of a project. Some will be executed by software tools; others will consist of human thinking, writing, exchanging ideas, and taking decisions through formal and informal interactions between members of the project team. All these activities must be supported by the process model. The process model enactment domain is concerned with the features needed to support process performance governed by the process model. These features support, guide, or enforce performance of the process in a way consistent with the process model.

The three domains interact with each other in different ways. Firstly, the process model influences the way in which the process is performed. Actual performance then corresponds 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 actual 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, thereby allowing its improvement.

This leads to a view of the usage world as imposing *strong requirements* on the way processes will be performed, the nature of process models used and the way in which these process models will be developed and changed. The purpose assigned to

the process model has to be determined by the usage world. This is captured below in the facet, *Purpose*. Since the way processes are performed changes with time, it is the duty of the organisation to define their process management policy. This is captured in the facet, *Process Management Policy*.

4.1 PURPOSES

A synthesis of proposals from the software engineering field [Lon93], [Cur92], the information system community [Bri90], [Pra97], [Rol96a], and the system design community [Ram92], [Pot89], show three main aims of process models:

- *descriptive*, to record trace what actually happens during a process,
- *prescriptive*, to define desired processes and how they should/could/might be performed,
- *explanatory*, to provide explanations about the rationale of processes.

A *descriptive* purpose takes the point of view of an external observer who looks at the way a process has been performed and determines the improvements that have to be made to make it perform more effectively or efficiently.

The *prescriptive* purpose lays down rules, guidelines, and behaviour patterns which, if followed, would lead to the desired process performance. The prescriptive purpose lies in a range from *strict enforcement* to *flexible guidance*. In the former the performance of the process must follow the prescription whereas in the latter the prescription is such that it can accommodate a large number of ways in which the process can proceed. Guidance shifts the emphasis away from task performance to goal achievement. Therefore, there can be two types of guidance, point and flow [Sis97]. Point guidance provides help in the achievement of a given goal whereas flow guidance helps in identifying the next goal in order for the process to proceed.

The *explanatory* purpose is important in those processes where several possible courses of action are open and each of these has to be explored and evaluated based on rational arguments. Such traces establish an explicit link between processes and the requirements that they are to fulfil.

The *descriptive* and *explanatory* purposes have been accorded a lot of attention in the recent past. This is because of the need to keep track of process knowledge and to support change [Got94], [Ram92]. To take this to the extreme, it is difficult to visualise any process, strategic, tactical, or implementation (see Subject World), without a descriptive and/or explanatory purpose behind them.

Specifically, if prescription is to be provided to strategic processes, then flexible guidance is clearly more appropriate than process enforcement. This is because strategic processes are often creative and require human co-operation. This makes most software process models inappropriate for strategic processes because [Fin94] their basic property is enforcement of constraints (prescriptions and even proscriptions). However, in tactical or implementation processes of the Subject

World that follow plans relatively more strictly and which are less creative and mercurial, varying shades of process enforcement ranging from mechanical enforcement with limited guidance to complete automation may be found useful.

A process engineering approach can be classified according to the role it aims to play in the *facet* called *Purpose* which has the three following attributes :

*Prescriptiv*e: ENUM {enforcement, guidance}
Descriptive: BOOLEAN
Explanatory: BOOLEAN

4.2 PROCESS MANAGEMENT POLICY

Processes change with time and so do the process models underlying them. Thus, new processes and models may have to be built and existing ones improved. There is need to have a well-defined organisational policy to handle this change. This policy can either accept *change continuously* as it occurs or accept it as one-shot, *radical change*. Radical change applies in situations where organisations need to define a process management policy from scratch. The former applies when need is felt to harmonise heterogeneous process practices or when a bottom-up approach is systematically applied to move up in the levels of maturity in the CMM [Hum89] framework. .

Strategic processes are highly unstable. The process proceeds by analogy with other similar processes and reuses experience and knowledge of their stakeholders. This reuse is continuous and operates so long as the process lasts. It is today implicitly done by individual human agents performing the process but, perhaps, in future, it shall be necessary to have *reuse* as a process management policy of the organisation. However, it remains to be conclusively shown that process practice reuse is cost effective in an organisational setting.

The foregoing is captured by the two attributes *change* and *reuse* of the *Process Management Policy facet*

Change: ENUM{continuous, radical}
Reuse: BOOLEAN

5. THE SYSTEM WORLD

If the subject world is the world of processes then the system world is the one of their representations. The interest in this world is in

a) what is to be represented
b) at what level of abstraction
c) how is it represented
d) what properties should the representation have.

The *facet contents*, of the system world deals with (a), the *abstraction facet* deals with (b), the *description facet* deals with (c), and finally, the *modularization facet* captures the properties of the representation. We develop each of these below.

5.1 ABSTRACTION

Processes of the same nature are classified together into a process model. Thus, a process model is a description of a process at the type level. Since the process model is at the type level, a process is an instantiation1 of it. The same process model is used repeatedly for the development of many applications and thus, has many instantiations. As stated in section 4, one possible use of a process model is to prescribe "how things must/should/could be done" in contrast to the process itself which is really what happens. A process model is more or less a rough anticipation of what the process will look like. What the process shall, indeed, be will, however, be determined during actual system development.

A *process meta-model* is a description at the type level of a process model. A process model is, thus, an instantiation of a process meta-model. Obviously, a meta-model can be instantiated several times in order to define various process models. A process meta-model is at the meta-type level with respect to a process. It plays a role similar to a theory in the *Theory of Plans* [Wil83] from which plans can be generated (the process models) and executed (the processes).

The *abstraction facet* captures the levels at which the model is expressed and the corresponding attribute takes on values from the *enumerated domain* { type, meta-type}.

The well known models like the waterfall [Roy70] and spiral models [Boe88] are the type level whereas the Nature process theory [Rol95] is at the meta-type level.

5.2 CONTENTS

The concern of this facet is with the contents of the process model/meta-model. These contents are determined by the system of concepts in terms of which processes are represented and by the granularity of these representations. These two aspects are dealt with by the *coverage* and *granularity* attributes respectively.

---

[1] A. Finkelstein in (Fin94) points out the various meaning of the widely used term "instantiation" in the software engineering community. We relate here to the classical idea of creating instances from a type/class definition

5.2.1 Coverage

According to Dowson [Dow88], process models can be classified into three groups of models:
- activity-oriented,
- product-oriented, and
- decision-oriented.

Since this classification was made, a new group called the contextual model has also emerged.

Activity-oriented

The *activity-oriented models* concentrate on the activities performed in producing a product and their ordering. These process models are sequential in nature and adopt the Cartesian, functional decomposition approach. They provide a frame for manual management of projects developed in a linear fashion. The first widely used process model, the Waterfall model [Roy70], falls into this category. Its widespread acceptance has led to life-cycle descriptions being most often treated as linear sequences where crucial aspects of the process such as feedback loops and iteration are not represented [Boe88], [Cur88] and [Cur92].

These models are well suited to model implementation processes. The strong emphasis on an activity incurs some risks of neglecting the influence of product structure on the process. Further, they are unable to support flexible prescriptive guidance but only process model enforcement. The linear view of activity decomposition seems inadequate to model creative processes because it is not possible to consider all contingencies. Activity-oriented representations cannot incorporate the rationale underlying the process and therefore do not permit reasoning about engineering choices based on existing conditions. It is unrealistic to plan what will happen in such a process in an entirely sequential manner. Finally, the linear view is inadequate for process models which have to support backtracking, reuse of previous designs and parallel engineering.

Product-oriented

Product-oriented process models, in a manner similar to activity-oriented ones, are centred around the notion of activity but, additionally, link activities to their output : the product. The ViewPoints model [Fin90] and the process model proposed in the European Software Factory (ESF) project [Fra91] belong to this category.

Product-oriented models couple the product state to the activities that generate this state. They visualise the process as a state transition diagram. Since product-oriented models adopt the notion of an activity, they suffer from the same difficulties as the activity-oriented models considered above. However, due to their product-activity coupling they are useful for tracing the transformations performed and their resulting products. However for strategic processes it is difficult, if not impossible, to write down a realistic state-transition diagram.

Decision-oriented

The successive transformations of the product caused by a process are looked upon, in decision-oriented models, as consequences of decisions. The process models of the DAIDA project [Jar92], [Pot89] and [Ram92] fall into this category. These models emphasise the concept of an "Intention " at the expense of an activity.

Decision-oriented models can be used for both, strategic as well as tactical processes. The strength of the decision-oriented approach is its ability to cover more aspects of a process than can be done by the two other kinds. Decision-oriented models are not only able to explain how the process proceeds but also why it proceeds. Therefore, decision-oriented process models are particularly well suited to strategic processes, for supporting explanatory tracing and prescriptive guidance. This is because of their ability to (a) guide the decision making process (b) help in reasoning about the rationale behind decisions,(c) support the deliberation underlying the decision process itself and (d) keep a trace of the happenings of a process and their rationale.

Contextual Models

Contextual models as found in the Nature process theory [Bub94], and in the F3 project [Rol94b] look upon each process as being in a subjectively perceived *situation* upon which is looked upon with some specific *intention.* The work to be done next depends on both the situation and the intention i.e. it depends on the context existing at this moment.

Contextual process models strongly couple the context of a decision to the decision itself. It makes the notion of a context, the coupling of a *situation* and the *decision,* central to process modelling. Decisions are applied to the situation in which the process currently is, in order to change that situation to the desired new one. In this respect, the contextual approach has some similarity with the planning paradigm that has emerged from Artificial Intelligence and with projects based on the planning paradigm such as GRAPPLE [Huf89].

Since the contextual models adopt the notion of a decision, all the properties of decision-oriented models mentioned earlier are applicable to them. Further, due to the strong relationship between the situation and the decision, only those decisions which are appropriate in the situation at hand are of interest. This helps in focusing guidance, tracing and explanation to specific process situations.

Process models can be classified within the *facet Contents*, by giving values to the *attribute*, *coverage*,

Coverage: ENUM{activity, product, decision, context}

5.2.2 Granularity

Most traditional process models are large-grained descriptions of the product life-cycle. On the other hand, there are very fine-grained models. For example specifying that after a source code file is edited, it should be recompiled [Kai88]. Recently, hybrid formalisms that use different notations for large-grain and small-grain aspects of process such as PROCESS WEAVER [Fer91], have been developed.

The nature of granularity needed is dependent on the situation at hand. Granularity affects the kind of guidance, explanation and trace that can be provided. High granularity limits these to a rather coarse level of detail whereas fine granularity provides more detailed capability. Process models should, ideally, provide a wide range of granularity. This shall allow a movement from large grains to fine grains along a continuum.

Therefore, *the granularity attribute* takes on values from SET(ENUM{large, fine, variable}).

5.3 THE DESCRIPTION FACET

The *description facet* is concerned with the form of the process representation and the level of formality of the language used to describe the process model. These correspond to the *form* and *notation attributes* of this facet.

5.3.1 Form

The form attribute is concerned with style of the process representation. There are three identified forms, *scripts*, *programs*, and *hypertext*.

Osterweil [Ost87] has proposed the view that software process models should take the form of a program as different from process scripts. Process scripts are interactively used by humans as against process programs which are enacted by a machine [Leh87]. They support non determinism whereas process programs can, at best, support process deviation under pre-defined constraints [Cug96].

The hypertext style of process representation is a network of links between the different aspects of a process, such as product parts, decisions, arguments, issues, etc.

A relationship can be established between *form* and the *purpose* facets of the Usage World. Scripts and programs are two styles which may be applicable to prescriptive purposes whereas hypertext is well suited to descriptive and explanatory purposes. Strict enforcement of the prescriptive purpose can clearly be represented in process programs whereas flexible guidance requires the process model to be represented in process scripts. Descriptive and explanatory purposes require the establishment of

relationships between different elements of a process trace. These relationships are well articulated as hypertext links.
The *form attribute* of the description facet takes on values from ENUM{script, program, hypertext}

5.3.2 Notation

Process models underlying information systems practice have traditionally used informal notations such as natural languages or diagrams with informal semantics. On the other hand, in software engineering, more formal software process models (see [Arm93], [Cur92], [Fin94] for an overview) have been used. This formality relates to underlying programming languages : Smalltalk for E3 [Fin94], various Prolog dialects for EPOS [Jac92], Oikos [Amb91], and PEACE [Fin94], PS-Algol for PWI [Fin94].

A formal notation is required to support the verification of the expected properties of the process model and validation of the process model using for instance, simulation or enactment techniques. The use of informal notations has made it difficult for process models to be followed systematically. Formal or semi-formal notations make these efforts considerably more effective. Formal notations are necessary for providing automatic enactment support.

The *notation attribute* helps classifying process models by one of the three values of the following enumeration:

Notation: ENUM{formal, semi-formal, informal}

5.4 MODULARIZATION

Early processes were monolithic. However, there is a shift towards modular process structure in this decade. We introduce a Boolean valued attribute called *Presence* in the modularization facet to distinguish between monolithic and modular methods.

One proposal for modularization [Har94] is that of fragments. A fragment can be either a *product fragment* or a *process fragment*. The drawback of the fragment based approach is the over-emphasis on the product fragment resulting in under developed meta-process modelling.

The proposal of [Rol93], [Rol94a], is to tightly couple the product and process aspects of processes into contexts. 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.

Process modules can be looked upon according to *two other perspectives* : *abstraction* and *aggregation*. Rolland [Rol95] has defined aggregates called process chunks as hierarchies of contexts. A chunk prescribes the way to proceed in the situation identified by the context at the root of its context hierarchy. This allows the decision of the root context to be taken in this situation. [Van96] proposes two kinds of aggregated modules called *route map* and *fragments* respectively. A *route map* refers to strategies such as delivery strategies, developmental strategies, realisation strategies etc., activities and products concerning system development as well as project management. The *fragment* is a coherent part of a process for system development or project management. Fragments may be linked to a route map which may establish a complete project approach.

Abstraction is used to capture generic laws governing the construction of different but similar process modules. Generic process modules can take the form [Rol96a] of *framework* or *pattern.* A framework models the commonality between modules of different process models but for the same type of application. A pattern models a common behaviour in process construction. It is generic in the sense that it is used every time a process model is constructed. Both terms have been chosen by analogy with reuse approaches in the object oriented area. Patterns are there defined as solutions to generic problems which arise in many applications [Gam93], [Pre95] whereas a framework is application domain dependent [Wir90], [Joh88].

Classification along the *modularization facet* comes down to giving values to the two following attributes:

Presence: BOOLEAN
Nature: SET(ENUM{primitive, aggregate, generic}

6. THE DEVELOPMENT WORLD

The *development world* deals with two issues
- the *process of constructing process models*, and
- *enactment of processes*.

The process of constructing process models is a *meta-process*, it is the process behind the process used to construct processes for building information systems products. The development world deals with meta-processes so as to improve process models and to make them evolve.

The second issue is that of *process enactment*. The development world is also concerned with the way in which process models can be constructed and process enactment support provided. That is, the tool environment needed to support process performance is also a concern of this world. Thus, the facets of interest in this world are *construction approach, construction technique, enactment support,* and *change support* .

6.1 CONSTRUCTION APPROACH

In a manner analogous to that of Harmsen [Har94] one can organise construction approaches in a spectrum ranging from 'low' flexibility to 'high'. At the 'low' end of this spectrum are rigid approach whereas at the 'high' end is modular approach. *Rigid approaches lead to process models that* are completely pre-defined and leave little scope for adapting them to the situation at hand. On the other hand, *contingency approaches* allow the modification and augmentation of models to make them fit to a given situation.

There are at least two ways by which contingency approaches can be realised. The first one is the *production of contingency process models* that is, situation-specific models for certain types of organisational settings. This presents process engineering as the selection of a model within a panel of contingency process models. In the second one process engineering is used to support the selection and the assembly of process components to construct *process models 'on-the-fly'*.

The foregoing suggests that *construction approach* should be classified as :

Construction approach: ENUM{contingency, on-the-fly, rigid}

The construction approach adopted in the development world has a strong impact on the *modularization* facet and *granularity* attribute of the system world. Whereas the rigid approach can be associated to monolithic models, contingency and on-the-fly approaches require modular process models. The contingency approach is well suited to support capitalisation of 'good practice' into process chunks in a stable environment. Instead 'on-the fly' approaches are adapted to the dynamic recognition and use of chunks and patterns.

6.2 CONSTRUCTION TECHNIQUE

Within the broad construction approach adopted for constructing process models, a number of techniques for construction are available. Construction techniques used in the information systems area have developed independently of those in software engineering. In information systems, construction techniques exploit the notion of a meta-model and the two principal techniques used are those of instantiation and assembly. In software engineering the main construction technique used today is language-based. However, early techniques in both, information systems and software engineering were based on the experience of process engineers and were, therefore, ad-hoc in nature. We comment the attributes values in turn.

6.2.1 Instantiation

Given that new process models shall be constructed very often, the question is how we can increase the productivity of process engineers and improve the quality of the models they produce. One way of doing this is to identify the common, generic features of process models and represent them in a system of concepts. Such a

representation has the potential to 'generate' all process models that share these features. This potential is realised when a generation technique is defined whose application results in the desired process model. Thus, there are two key issues here
- the identification of the system of generic concepts
- the instantiation technique.
The first of these is resolved by the definition of a *process meta-model* whereas the second issue is resolved by deriving process models from this process meta-model through *instantiation*. A number of advantages flows from this:

1) The exploitation of the meta-model helps us to define a wide range of process models.

2) It makes the activity of defining process models systematic and versatile.

3) It forces us to look for and introduce, in the process meta-model, generic solutions to problems and this makes the derived process models inherit the solution characteristics. Under the instantiation approach, the crucial issue in process modelling is no longer the process model but the process meta-model. This means that the onus of providing a process model with the required characteristics shifts from the process model developer to the process meta-model developer.

The instantiation technique has been used, for example, in NATURE [Rol93], [Rol94], [Rol94a], [Rol96a]. The process engineer must define the instances of contexts and relationships that comprise the process model of interest. It has been utilised to build the repository of Computer Aided method Engineering environments [Kel96], [Har95], [Mer91], [Sis96].

6.2.2 Language

The software engineering community has used different languages for expressing process models like Smalltalk for E3 [Fin94], various Prolog dialects for EPOS [Jac92], Oikos [Amb91], and PEACE [Fin94], PS-Algol for PWI [Fin94]. Different computational paradigms have also been used, for example, Petri nets in EPOS [Jac92] and SPADE [Ban93], rule based paradigm in MERLIN [Emm91], ALF [Ben89], Marvel [Kai88], EPOS [Jac92], and triggers in ADELE [Bel89] and MVP-L [Fin94].

There is a relationship between the construction technique and the form facet in the system world. Indeed, languages are typically related to process programs whereas instantiation techniques have been used to construct process scripts.

6.2.3 Assembly

The assembly technique relies on the availability of process components in a process repository. Assuming that process components exist in a process repository, the question now is "how to deliver the relevant process components to the user?" The process 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 process components support local matching with the situation at hand.

[Van96] is an example of the first approach. This approach has been tried out in nine non-standard projects of the systems development department of a bank organisation. The second approach [Rol96b] uses the notion of descriptor [DeA91] as a means to describe process chunks. It has been tried out to construct information systems methods [Pli95] in NATURE and repository of scenario based approaches accessible on Internet in the CREWS project [Rol98].

For the assembly technique to be successful, it is necessary that process models are modular. If the assembly technique is combined with the instantiation technique then the meta-model must itself be modular.

6.2.4 Ad-Hoc

Traditional process models are expressions of the experiences of their developers. Since this experience is not formalised and is, consequently, not available as a fund of knowledge, it can be said that these process models are the result of an ad-hoc construction technique. This has two major consequences : it is not possible to know how these process models were generated, and they become dependent on the domain of experience. If process models are to be domain independent and if they are to be rapidly generable and modifiable, then we need to go away from experience based process model construction. Clearly, generation and modifiability relate to the process management policy adopted (see Usage World). Instantiation and assembly, by promoting modularization, facilitate the capitalisation of good practice and the improvement of given process models.

The construction technique facet is defined as follows:

> *Construction technique:* SET(ENUM{instantiation, language, assembly, ad-hoc})

6.3 ENACTMENT SUPPORT

Enactment mechanisms have been mainly implemented by the software engineering community as the core of Process Centred Software environments. An enactment mechanism determines and controls the interactions between the agents performing the process so as to trace, guide, and enforce performance of the process in a way consistent with the process model.

Considerable effort has been put in to provide automated execution support, automated monitoring and enforcement of software processes in process centred software environments. The reader will find in [Fin94] a detailed presentation of ten projects in the field as well as the results of a common assessment exercise performed by the leaders of these projects.

Most process centred software environments [Jac92], [Bel94], [Ban93], [Kai88] are in fact used to describe the activity of tools and to allow automatic invocation of tools [Tom94]. Existing environments guide software engineers in the selection of the right suite of tools but they do not guide the engineering activities themselves. On the contrary, some attempts have been made in the information systems community for implementing enactment mechanisms that focus on guiding engineering activities [Sis96]

Whereas the foregoing deals with supporting the performance of application processes, there is also need to support the process of constructing process models, the meta-process. Just as other processes are represented by process models, the meta-process shall have its own model, the meta-process model. Again, the meta-process itself needs to be performed in accordance with the meta-process model and this means that enactment support has to be provided for the performance of the meta-process.

It is possible to build two separate enactment environments for dealing with process and meta-process enactment respectively. However, if the *meta-process* is treated as just another process then it is possible to use the same enactment mechanism to support both, the process and the meta-process. In fact this has been demonstrated in the Mentor environment[Sis96].

Enactment mechanisms must support processes that take the form (see System World) of scripts, programs, or hypertext. When a process model is a script then the enactment mechanism provides high flexibility so as to enable human agent intervention during process performance. This intervention would be supported by guidance mechanisms which may either, proactively provide suggestions on possible decisions that could be taken or may support requests for help. In terms of the Usage World, for models which are process programs, the enactment mechanism behaves like the run-time support of programming languages. Process program enactment is program execution whereas process script enactment is model interpretation. Finally, when models are of the hypertext form then the enactment mechanism offers facilities to create links between process decisions, their rationale, and the resulting product.

Since the meta-process is a process, it is possible to extend the foregoing remarks to it as well. However, as it is unlikely to completely plan out the meta-process, it would be the case that meta-process models are not treated as process programs but as process scripts only.

This facet has two Boolean values *attributes*

Process support: BOOLEAN
Meta-process support: BOOLEAN

6.4 CHANGE SUPPORT

The traditional practice is that if a process model does not meet requirements of the users then a new one is built. This practice causes loss of experimental knowledge which could have been used to change process models. The development world must therefore provide support for *process model change*.

There are two different ways in which this can happen
(a) process model change takes place even as the process proceeds: the process model can be adapted to specific requirements as these emerge,
(b) the process model may need to be revised and improved at the end of the project: this is to benefit from the experience gained in process model use.
The former is referred to as *dynamic process change* [Dow94] and the latter as *process improvement* [Lon93].

Different positions are taken in the software process engineering community concerning the need for dynamic changes. On the one hand, people claim that this is an essential requirement and some software process centred environments EPOS [Jac92], E3 [Fin94], SPADE [Ban93], ADELE [Bel89] try to provide solutions for it [Fin94]. On the other hand, it can be argued that a prescriptive approach to process modelling is at odds with dynamic process evolution [Hum92]. The notion of *fitness of the process* has been defined in [Hum92] as the degree to which the agents performing the process can faithfully follow the process steps it specifies. When this fitness is low then process change occurs. Thus, process model change is an indication of lack of flexibility of the model. Recent process models include the concept of process deviation and therefore control the deviation of process performance from that prescribed in the process model.

There are only initial experiments in improving the process model by experience-based learning, as suggested in the literature [Hum89], [Oiv92], [Poh92]. They suggest two ways of process improvement, by inductive or deductive learning. Inductive learning is based on the analysis of process deviations that are recorded in process traces. Induction improvement can be performed by a human agent who, on his own, decides the change that is needed. The agent can be supported by generalisation rules [Mic83] that can be part of a tool based inductive learning environment [Pra96]. In order to do inductive improvement, there must exist a mapping between elements of the trace and the concepts of the process model.

Deductive learning exploits Case-based reasoning. Thus, it solves new problems by adapting solutions that have been utilised to resolve past problems [Rie89]. Case based reasoning when applied to process performance calls for the existence of a repository of cases. Deductive process improvement aims at adding new cases in the repository by examining process performances. Deductive learning corresponds to the retaining phase of the Case based reasoning cycle which traditionally consists of the four phases (a) retrieve, (b) reuse, (c) revise, and (d) retain.

Dynamic process change and process improvement are the two techniques that the Development World can offer to support the process management policies set in the Usage world. Deductive process improvement is appropriate when an organisation wants to promote the reuse of good practice in performing processes. Clearly, a process model supporting aggregates (see Modularization facet in System World). shall be well suited to provide these as cases to be stored in the repository.

Inductive improvement is well suited to situations where process models are used repeatedly and can continuously be improved by learning from the deviation of actual performances. A modular structure of process models helps in relating the observed deviations to specific, localised parts of the process model components and therefore facilitate inductive improvement.
The change support attribute takes one or several values among the following enumerated domain :

> *Change support*: SET(ENUM{dynamic process change, process improvement})

7. CONCLUDING REMARKS

The subject and the usage worlds constitute the environment within which the technical view of process engineering contained in the system and development worlds lies. This embedding is the source of the inter-relationships between the facets of the four views discussed in this paper.

The nature of processes and the purpose imposed by the usage world on the process engineering solution determine, to a large extent, the type of contents and description of the process models/meta-models. The choice of a particular type of content and description based on the nature of the processes guarantees that the semantics of these processes are well captured in process models/meta-models. On the other hand, selection of a content and description to meet the purpose expressed by the usage world guarantees that the process model/meta-model shall fulfil the requirements of the process stakeholders. In fact, we suggest that selection based on purpose should have primacy over that based on the nature of the process. This conclusion can be drawn by analogy with ISE approaches where it has been recognised that user needs are better met by understanding usage goals and not merely by using good semantic conceptual model.

In fact, the usage world affects the system world through both, the purpose as well as the process management policy. Of the three purposes, the explanatory and the descriptive have been incorporated in process models that provide design rationale and process traceability respectively. However, the issue of providing prescriptive guidance is still open. The process management policy affects the abstraction facet as it introduces the need for abstracting process models into process meta-models. Before building meta-models which reflect old models but on a new level of abstraction, one should question the old ones. The goal of meta-modelling is not

only to operationalise current process models but also to correct the general oversights and limitations of these.

In a similar manner, we believe that the technical solution in the development world has to be chosen according to the purpose and the process management policy decided in the usage world. The influence of the former is clearly on the choice of the enactment mechanisms. The implication of the latter is more diverse. The policy recognises the need for managing processes in-the-large and their evolution in time. The former sets the requirements of an automated enactment support and its extension to the meta-process. The latter determines the choice of the change support and of the construction approach. There are two implications of this, organisational and technical. The absence today of organisation-wide process policies raises the need for organisations to understand the crucial role played by these policies and to define them. Such policies would, for example, encourage capitalisation of good practice, learning from experience, and development of a reuse culture.

The capitalisation policy raises the technical question of how good practices can be recognised, organised and reused. Such knowledge should be available in the form of chunks for later assembly. This implies the modularization of process models/meta-models. The modular approach represents a shift in the way of thinking about process representations and is likely to emerge as a major research issue in the future.

The reuse culture raises the question of the genericity of process representations. Perhaps, what is needed is a corpus of both, generic process and generic meta-process knowledge in a modular form. A suggested research is therefore, the development of a domain analysis approach to identify common properties of (a) different process models and (b) different meta-process models.

The evolution of process models calls for the establishment of technical enactment support for the meta-process. So far, work has concentrated on developing and experimenting with process enactment mechanisms. The research issue here is of making these mechanisms generic enough to handle both process and meta-process enactment.

This paper has argued that process engineering should be usage driven. The acceptance of process engineering in organisations is, however, not entirely determined by the functionality that is needed but also by other non-functional factors such as usability, availability, security etc. This aspect of process engineering has to be addressed by the research community more vigorously.

8. REFERENCES

[Amb91] : V; Ambriola, M. L. Jaccheri, *Definition and Enactment of Oikos software entities*, Proc. of the First European Workshop on Software Process Modeling, Milan, Italy, 1991

[Arm93] 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.

[Ban93] S. Bandinelli, A. Fugetta, S. Grigoli, *Process Modelling in the large with SLANG*, Proc. of the 2nd Int. Conf. on Software Process, Berlin, Germany, 1993, pp 75-93.

[Bel94] N. Belkhatir, W. L. Melo, S*upporting Software Development Processes in Adele2*, in the Computer Journal, vol 37, N°7, 1994, pp 621-628..

[Ben89] K. Benali, N. Boudjlida, F. Charoy, J. C. Derniame, C. Godart, Ph. Griffiths, V. Gruhn, Ph. Jamart, D. Oldfield, F. Oquendo, *Presentation of the ALF project*, Proc. Int. Conf. on System Development Environments and Factories, 1989.

[Boe88] B. Boehm, *A Spiral Model of Software Development and Enhancement*, IEEE Computer 21, 5, 1988.

[Boeh76] B. Boehm, *Software Engineering*, IEEE Transactions on Computers, Vol. C-25, No. 12, 1976.

[Bri90] S. Brikemper, Formalisation of information systems Modelling, Ph. D. Thesis, University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

[Bub94] J. Bubenko, C. Rolland, P. Loucopoulos, V. De Antonellis, Facilitationg *Fuzzy to Formal Requirements Modelling*, In the Proc. of the 1st ICRE Conf., Colorado Springs, USA, April, 1994

[Cug96] G. Cugola, E Di Nitto, A. Fuggetta, C. Ghezzi, A farmework for formalizing *Inconsistencies and deviations in human centred systems*, ACM Transactions on software engineering and methodology (TOSEM), Vol 5, N° 3, July 1996.

[Cur88] B. Curtis, M. Kellner, J. Over, *A Field Study of the Software Design Process for Large Systems*, Com. ACM, Vol. 31, No. 11, 1988.

[Cur92] B. Curtis, M. Kellner, J. Over, *Process Modeling*, Communications of ACM, vol 35 n°9, september 1992, pp 75-90.

[DeA91] 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

[Dow88] M. Dowson, *Iteration in the Software Process*, Proc 9th Int. Conf. on Software Engineering, 1988.

[Dow93] M. Dowson, *Software Process Themes and Issues*, IEEE 2nd Int. Conf. on the Software Process , pp 28-40, 1993.

[Dow94] M. Dowson, C. Fernstrom, *Towards requirements for Enactement Mechanisms*, Proc. of the th European Workshop on Software Process Technology, 1994

[Emm91] : W. Emmerich, G. Junkermann, W Schafer, *MERLIN : knowledge-based process modeling*, Proc. of the First European Workshop on Software Process Modeling, Milan, Italy, 1991.

[Fer91] C. Fernström, L. Ohlsson, *Integration Needs in Process Enacted Environments*, Proc. 1st Int. Conf. on the Software Process, IEEE computer Society Press, October 1991.
[Fin90] : Finkelstein A. , Kramer J. , Goedicke M. : *ViewPoint Oriented Software Development*; Proc. Conf Le Génie Logiciel et ses Applications, Toulouse, p 337-351, 1990.
[Fin94] A. Finkelstein, J. Kramer, B. Nuseibeh (eds), *Software Process Modelling and Technology*, John Wiley (pub), 1994.
[Got94] O. C. Z. Gotel, A. C. W. Finkelstein, *An analysis of the requirements traceability* problem, In Proc. Of Int. Conf. On Requirements engineering, ICRE'94.
[Fra91] M. Franckson, C. Peugeot, *Specification of the Object and Process Modeling Language*, ESF Report n° D122-OPML-1. 0, 1991.
[Gam93] 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
[Har94] : Harmsen A.F., Brinkkemper J.N., Oei J.L.H.; *Situational Method Engineering for information Systems Project Approaches*, Int. IFIP WG8. 1 Conf. in CRIS series : Methods and associated Tools for the Information Systems Life Cycle (A-55), North Holland (Pub.), 1994.
[Har95] Harmsen F., Brinkkemper S., *Design and implementation of a method base management system for situational CASE environment*. Proc. 2nd APSEC Conf., IEEE Computer Society Press, pp 430-438, 1995
[Huf89] : K. E. Huff, V. R. Lessor, *A plan-based intelligent assistant that supports the software development process*, Proc. of the 3rd Software Engineering Symposium on Practical Software Development Environments, Soft. Eng. Notes, 13, 5, 1989, pp97-106
[Hum89] Humphrey, W. S. : *Managing the Software Process*, Addison-Wesley, 1989. (verifier CMM)
[Hum92] Humphrey W. S, P. H Feiler, *Software Process Development and Enactment : Concepts and Definitions*, Tech. Report SEI-92-TR-4, SEI Institute, Pittsburgh, 1992
[Jac92] L. Jacherri, J. O. Larseon, R. Conradi, *Sotware Process Modelling and Evolution in EPOS*, in Proc. of the 4th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'92), Capri, Italy, 1992, pp574-589.
[Jar92] M. Jarke, J. Mylopoulos, J. W. Schmidt, Y. Vassiliou, *DAIDA - An Environment for Evolving Information Systems*; ACM Trans. on Information Systems, Vol. 10, No. 1, 1992.
[Jar93] M. Jarke, K. Pohl, *Requirements Engineering: An Integrated View of Representation, Process and Domain*, Proc. 4th European Software Conf., Springer Verlag, 1993
[Jar94] M. Jarke, K. Pohl, C. Rolland, J. R. Schmitt, *Experienced-Based Method Evaluation and Improvement : A Process Modeling Approach*, Int. IFIP WG8. 1 Conf. in CRIS series : Method and associated Tools for the Information Systems Life Cycle, North Holland (Pub.), 1994.

[Joh88] Johnson R. E., Foote B., *Designing reusable classes*, Journal of Object-Oriented Programming, Vol 1, No3, 1988
[Kai88] G. E. Kaiser, N. S. Barghouti, P. H. Feiler, R. W. Schwanke, *Database Support for Knowledge-Based Engineering Environments*, IEEE Expert, 3(2), 1988, pp18-32.
[Kel96] Kelly S., Lyyttinen K., Rossi M., *Meta Edit+: A fully configurable, multi-user and multi-tool CASE and CAME environment*, Proc. CAiSE'96 Conf., Springer Verlag, 1996
[Leh87] M. M. Lehman, *Process models, process programming, Programming support*, Proccedings of the 9th Int. Conf. on software engineering, Monterey, California, USA, 1987
[Lon93] J. Lonchamp, *A structured Conceptual and Terminological Framework for Software Process Engineering*, Proc. Int Conf. on Software Process, 1993
[Lub93] M. Lubars, C. Potts, C. Richter, *A Review of the State of the Practice in Requirements Modeling*, Proc. Int. Symposium on Requirements Engineering, 1993.
[Mer91] Merbeth G., *Maestro II- das intergrierte CASE-system von Softlab*, CASE systeme and Werkzeuge (Ed. H. Balzert) BI Wissenschaftsverlag, pp 319-336,1991
[Mic83] R. S Michalski, *A Theory and Methodology of Inductive Learning*, Atificial Intelligence, Vol 20, No 2, 1983
[Oiv92]M. Oivo, V. R. Basili, *representing software engineering model : the TAME goal oriented approach*, IEEE Transactions on Software Engineering, Vol. 18 , N° 10, 1992.
[Oll88] T. W. Olle, J. Hagelstein, I. MacDonald, C. Rolland, F. Van Assche, A. A. Verrijn-Stuart, *Information Systems Methodologies : A Framework for Understanding*, Addison Wesley (Pub.), 1988.
[Ost87] L. Osterweil, *Software processes are software too*; Proc. 9th Int. Conf. on Software Engineering, IEEE Computer Societ, Washington, DC, 1987, pp2-13
[Poh92] K. Pohl, *Quality information systems : Repository for eveloving process models, Aachen Informatik, Beichte 92-37-RWTH, Aachen*.
[Poh93] K. Pohl, *The three dimensions of Requirements engineering*. In Proc. of the 5th Int. Conf. on advanced Information Systems Engineering, pp. 275-292, Paris, France, June 1993. Springer-Verlag.
[Pli95] V. Plihon, C. Rolland, *Modelling Ways-of-Working*, Proc 7th Int. Conf. on Advanced Information Systems Engineering (CAISE), Springer Verlag, 1995.
[Pot89] C. Potts, *A Generic Model for Representing Design Methods*, Proc. 11th Int. Conf. on Software Engineering, 1989.
[Pra96] : N. Prat, *Using Machine learning techniques to Improve Information Systems Development Methods*, 2nd AIS Americas Conf. on Information Systems, Phoenix, USA, 1996.
[Pra97] N. Prakash, *Towards a formal definition of methods, in Requirements Engineering*, Vol. 2 , N° 1, 1997.
[Pre95] Pree W., Design Patterns for Object-Oriented Software Development, Addison Wesley, 1995
[Pri87] R. Prieto-Diaz, P. Freeman, *Classifying Software for reusability*, IEEE Software, Vol. 4, N° 1, January 1987.

[Ram92] B. Ramesh, V. Dhar, *Supporting Systems Development by Capturing Deliberations During Requirements Engineering*, IEEE Trans. on Software Engineering, Vol 18, No6, 1992.
[Rie89]. C. riesbeck, R. Schank, *Inside Case-based Reasoning*, Erlbaum(ed.), Northvale, New Jersey, USA, 1989
[Rol93] C. Rolland, *Modeling the Requirements Engineering Process*, Information Modelling and Knowledge Bases, IOS Press, 1993.
[Rol94a] : Rolland C., *A Contextual Approach to modeling the Requirements Engineering Process*, SEKE'94, 6th Int. Conf. on Software Engineering and Knowledge Engineering, Vilnius, Lithuania, 1994
[Rol94a] Rolland C., Grosz G., *A General Framework for Describing the Requirements Engineering Process*, C. IEEE Conf. on Systems Man and Cybernetics, CSMC94, San Antonio, Texas, 1994
[Rol94b] C. Rolland, *Modelling the evolution of artifacts*, In Proc. of the first Int. Conf. on Requirements Engineering, April, 1994.
[Rol95] C. Rolland, M. Moreno, C. Souveyet, *An approach for beginning ways of working*, In Information System Journal, Vol. 20, N° 4, 1995.

[Rol96a] 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
[Rol96b] C. Rolland, N. Prakash, A proposal for context-specific method engineering, IFIP WG 8.1 Conf. on Method Engineering, Chapman and Hall, pp 191-208, 1996.
[Rol97] C. Rolland, *A Primer For Method Engineering*, In Actes du congrès Inforsid 97, Toulouse, France, June 1997.

[Rol98] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, M. Jarke, P. Haumer, K. Pohl, Dubois, P. Heymans, *A proposal for a scenario classification framework*. To appear in Requirements Engineering Journal 3 :1, 1998.
[Ros91] T. Rose, M. Jarke, M. Gocek, C. Maltzahn, H. W. Nissen, *A Decision-based Configuration Process Environment*, IEEE Software Engineering Journal, 6, 3, 1991
[Roy70] Royce W. W. : *Managing the Development of Large Software Systems*; Proc. IEEE WESCON 08/1970
[Sis96] S. Si-Said, C. Rolland, G. Grosz, *MENTOR :A Computer Aided Requirements Engineering Environment*, in Proc. of CAiSE' 96, Crete, GREECE, May 1996.

[Sis97] S. Si Said, *Guidance for requirements engineering processes*. Proc. of the 8th Int. Conf. and Workshop on Database and Experts System Application DEXA'97, Toulouse, 1-5 September 1997.
[Tom94] K. Tominaga, T. Tokuda, *Constraint-Centered Descriptions for Automated Tool Invocation*, IEEE Asia-Pacific Software Engineering Conf. (APSEC), 1994, pp92-101.
[Van96] K. Van Slooten, B. Hodes, *Characterising IS development project*, IFIP WG 8.1 Conf. on Method Engineering, Chapman and Hall, pp 29-44, 1996.

[Wil83] Wilenski, *Planning and Understanding*, Addison Wesley (Pub.), 1983.
[Wir90] Wirfs-Brock J., Johnson R., *Surveying current research in Object-Oriented Design*, Communications of ACM, Vol. 33, No9, 1990
[Fei93]P. H. Feiler, W. S. Humphrey, *Software Process Development and Enactment: Concepts and Definitions*, Proc. 2nd Int. Conf. on "Software Process", 1993.