# A GENERIC APPROACH TO SUPPORT A WAY-OF-WORKING DEFINITION

Mario Moreno, Colette Rolland, Carine Souveyet

Université de Paris 1 Panthéon-Sorbonne
Centre de Recherches en Informatique
17 Rue de Tolbiac, 75013 Paris
{ moreno | rolland | souveyet }@masi.ibp.fr

**Abstract.** Information System Engineering has made the assumption that an Information System is supposed to capture some excerpt of the real world history and hence has concentrated on systems modelling. Very little attention has been paid to the conceptual modelling process. However the emphasis on system modelling is shifting to process modelling. The particular process modelling approach being presented in this paper advocates the definition of a way-of-working (i.e. process models) to control and guide developers. The paper introduces a classification of the various kinds of evolution of objects and presents a decision-oriented process meta model to structure ways-of-working. We also describe some guidelines, related to our classification of object evolutions, to support method engineers in the task to define a way-of-working.

## 1 Introduction

All recent developments in the field of Software Engineering, Databases and Information Systems seem to show that support for the various stakeholders involved in Software projects can be provided by capturing the history about the design decisions, in the early stages of the systems development life cycle, in a structured manner. Much of this knowledge which we call the process knowledge is nowadays lost in the course of engineering and maintaining such systems. In large projects, in the course of time and with changing development groups, the rationale for and context of key design decisions are confused and even lost [3]. This is particularly true in Information System Engineering which has made the assumption that an information system (IS) is supposed to capture some excerpt of the real world history and hence has concentrated on systems modelling. It is usual to view an information system as "a model of some slice of the reality of an organisation" [9] and even to regard the IS development as a problem of models construction and description. This practice provides an answer of sorts to the fundamental question : what does the information handled by an information system means? It also tends to draw the attention away from another equally fundamental question : how to define which information has to be handled by an information system?

The emphasis on product[11] i.e. the system models has hidden the importance of process i.e. the route to deliver the product. A large variety of models and especially conceptual models by which an IS can be modelled in high level terms have been developed. In contrast, very little attention has been paid to the modelling process which has the purpose of investigating the requirements of the users community and abstracting from them the conceptual specification of the IS (namely, the Requirements Engineering (RE) process).

Recent research works (see for example the special issue of the IEEE Transactions on Software Engineering on Knowledge Representation and Reasoning in Software Development, Vol. 18, Nb 6, June 1992) converge to the central idea that process modelling is as important as system modelling is. For lack of process modelling, understanding of what the development process is, what happens during it, when, why, on what it happens, by whom it is performed, is very poor.
The process semantics are not well captured, with the required level of detail, in existing process models. Consequently, the way-of-working prescribed by methodologies is badly defined, and the derived CASE-tools are efficient in recording, retrieving and manipulating system models but are almost unable to actually support the developers in performing the creative activities of model construction and transformation. Controlling and guiding developers in the progressive elaboration of a product, tracing in a structured manner the history of an IS development, keeping track of each transformation of the product, of what occurred and when in order to improve a way-of-working definition by learning is almost not possible without an adequate process modelling approach.

In the F3[1] Esprit project the need for process modelling and tracing has motivated the process stream of the project. F3 aims at defining a methodology that integrates into a coherent whole a selection of adequate techniques, all aiming at improving requirements acquisition, elicitation and validation to facilitate the construction of the IS specification. Part of our contribution within this project is to define a process modelling approach in order to be able to provide a guidance tool and a trace tool to support the developers' tasks during the RE phase.

This communication is centred on a way-of-working definition. We introduced our process modelling approach and the underlying process meta model in section 2. In section 3, we introduce some guidelines to support method engineers in the task to define ways-of-working. Finally, to centre the work presented in this paper and conclude, the architecture for CASE-Tools that we have defined to completely support our process modelling approach is introduced.

## 2 Way-of-working modelling

The central thesis of this paper is that a way-of-working may be partially defined by instanciation of a process meta model. In this section, such a process meta model is introduced. It has been defined for the F3 Esprit III project as an extension of the process meta model developed in the NATURE Esprit III project for the Requirements Engineering (RE) phase.

### 2.1 Abstraction Levels in Process modelling

We distinguish three successive abstraction levels in process modelling (see figure 1) : *process* level, *process model* level and *process meta model* level.

A *process* is an organised set of both human and computerised activities which have led to a given product definition. In Requirements Engineering, a process describes what happens during the activity of specifying an Information System by abstracting from an initial set of users' requirements. The output of a process is also called product.

---

[1]Esprit III project (n°6612) named "From Fuzzy to Formal"

A *process model* describes any process resulting from the use of a given RE methodology. i.e. it describes how things should be or could be done for the purpose of RE products prescriptive building. As a matter of fact, a process model intends to formally specify the way of working prescribed within a RE methodology.
A *process meta model* describes the generic concepts required to define process models. It brings and relates the concepts allowing to characterise any way-of-working definition.

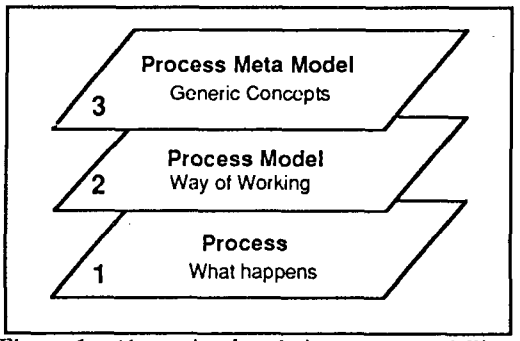


**Figure 1** : Abstraction Levels in process modelling

According to this view, a process model is an instanciation of a process meta model and a process is an instanciation of a given process model which is executed. This view is similar to the theory of plans [20] from which plans can be generated (the process models) and executed (the processes).

## 2.2 The Process meta model

Existing process meta models can be classified into three categories [4] : activity-oriented, product-oriented and decision-oriented. The NATURE process meta model [15] is an extension of the decision-based approaches (see figure 2).
Considering the highly non-deterministic nature of the Requirements Engineering process we believe that only the decision-based approaches appear to be partially appropriate, even if they offer limited hints about when and how to decide on what. It is probably impossible to write down a realistic state-transition diagram that adequately describes what happens in requirements engineering (activity-based approaches). But relying on a pure artefact history is also insufficient (product-based approaches). Analysts react *contextually* according to the domain knowledge they acquire and react by analogy with previous situations they have been involved in. Our process modelling approach aims at capturing not only activities performed during the RE process but also why these activities are performed (the decisions) and when (the decision contexts).

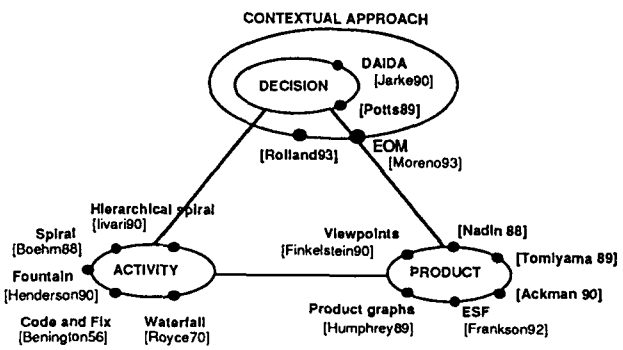


**Figure 2** : State of art in process modelling

Figure 3 gives an overview of the process meta model, introducing its key concepts and their relationships (with a binary E/R based notation). This process meta model is an extension of the NATURE meta model [15] which was also an extension of the one presented in [6] and implemented in the ALECSI prototype [14].

From a theoretical point of view the process meta model defines a set of generic concepts allowing to look upon any RE process model as a network of types :

- *situation* to explain the object(s) view on which it makes sense to take a decision on. A product meta model based on a detailed description of situations is introduced in [18] ;
- *decision* to reflect a choice that a requirement engineer could take in some pre-defined situations. A decision is looked upon as an intention of object evolution ;
- *action* to implement an object evolution. It is a materialisation of the decision. Performing it, at the process level, changes the current definition of an object and may generate new situations which, in turn, are subjects to new decisions ;
- *argument* to support and object to a decision ;
- *object* to refer a product element of a particular RE methodology.

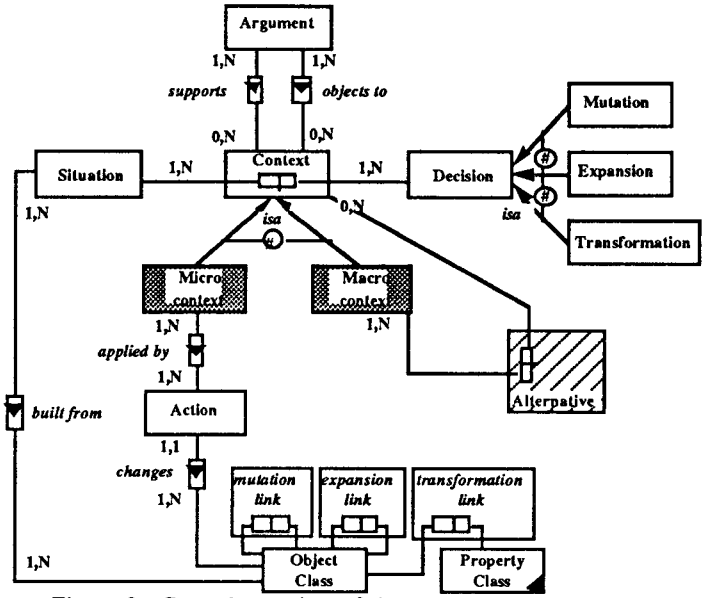


**Figure 3** : General overview of the process meta model

A *context* is a couple <situation, decision> which amplifies the decision semantics by refining the "when" part of the decision. Nevertheless, the process model has to handle different levels of granularity in decision making. A *micro context* refers to a situation in which the tactic to follow to perform the decision is defined. A *macro context* refers to a strategic *decision* that has to be refined before being implemented. A macro context corresponds to a situation with possible alternatives in decision-making. The decomposition of a macro context into its more elementary contexts is represented in Figure 3 by the loop upon context through macro context and alternative.

For instance the decision to improve a product is an intention which can be achieved through several alternatives. We show, in figure 4, three alternatives related to an

Entity_type *User* : *Improve_Entity_type*, *Improve_Entity_type_in_Role* and *Add_Card*. Both first alternatives are macro contexts which are in turn refined. A terminal leaf, in a context hierarchy, means that there are no alternatives defined for it i.e. there is only one procedure to apply this decision within this situation. < {User} ; Attributise > is such an example of what we call a micro context.

In addition the process meta model recognises three types of object evolution [16], [10] : *expansion, transformation and mutation.* This classification leads to the specialisation of decisions in three groups as well as a classification of objects relationships (see figure 3) : *expansion_link, transformation_link* and *mutation_link.*

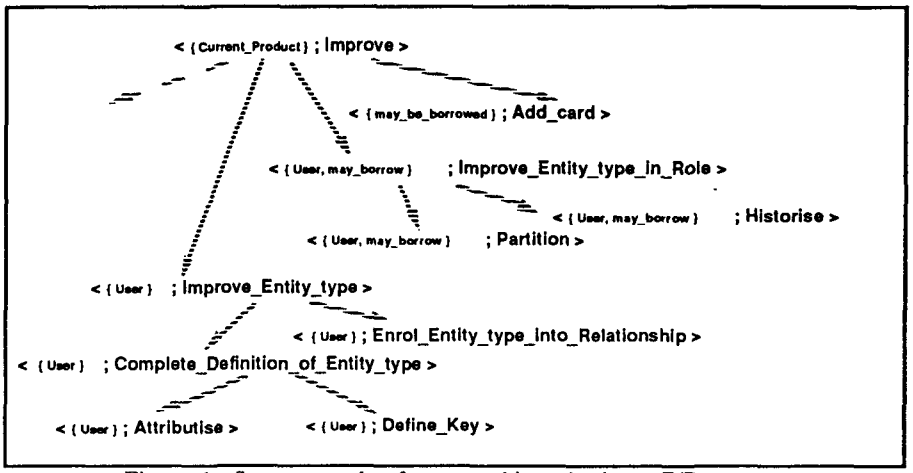


Figure 4 : Some example of contexts hierarchy in an E/R process

An *expansion* affects an expansion link between two Objects. These bi-directional links result of the structural relationships existing among the concepts of the models in use. For instance, any Entity-type may be expanded with attributes.
A *transformation* affects a transformation link between an object and a property. Such a link results of the existence of properties related to a concept. For instance, any attribute may have properties such as a name, a domain and its valuation (mono or multi).
A *mutation* occurs to an object when its type changes. This kind of link results of the mutation of concepts. For instance any relationship may be *retyped* as entity-type and further *mapped* into a relational table. By this way, an IS development may be partially viewed as a mutation process which abstracts from users requirements the conceptual specification of the information system and then converts it into an implemented system.

The mutation links are not precisely defined nowadays. We believe that they represent an important aspect of any way-of-working definition. We also believe that any way-of-working could be partially defined by a set of macro and micro contexts covering Engineering decisions on Product progressive building. In next section, we introduce an approach to facilitate this definition.

## 3 Way-of-working definition support

Stating that a way-of-working can be defined as a set of macro and micro contexts, we propose a generic decision hierarchy, based on the classification of object evolution

(expansion, transformation and mutation), to guide a method engineer in the definition of the required set of contexts for a methodology. The examples used in this section are part of the F3 way-of-working, which has been defined with this approach, and focus on the mapping from the Enterprise model [1] to the IS model [19].

### 3.1 Pre-defined set of generic decisions

In the previous section we have introduced a specialisation of decisions based on a three-dimensional view of object evolution : *expansion, transformation and mutation.* We propose here to pre-define a set of generic decisions for each class. This set is introduced in figure 5, each class of which is introduced in turn showing micro decisions (terminal leaves) and macro decisions (non-terminal leaves).

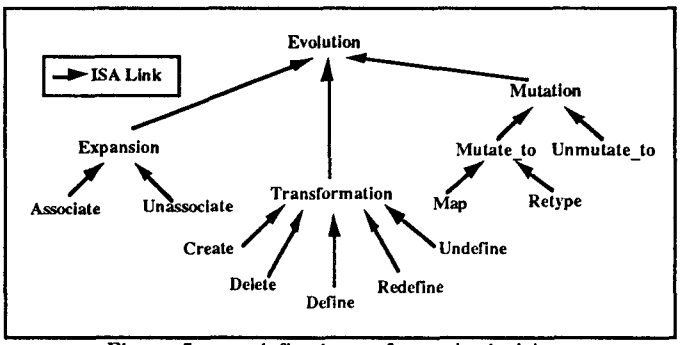


**Figure 5** : pre-defined set of generic decisions

The transformation decisions affect what we call the *Inner_environment* of an Object. This environment contains the object itself and its possible properties. We have identify five types of transformation decisions : *create, delete, define, redefine* and *undefine.*
*Create/delete* is the decision to instanciate/remove an object. *Define* allows to associate property to an object. Notice that the related action creates a property instance and also a transformation link between both object and property. *Redefine* allows to change an existing property. Finally, *undefine* allows to delete a property previously defined. The related action deletes both the transformation link and property.

The expansion decisions express how expansions may affect what we call the *Spatial_environment* of an Object. This environment contains the objects which may be structurally and directly connected to it. We have identify two types of expansion decisions: *associate* and *unassociate. Associate* is the decision to structurally relate two objects. The associated action creates an expansion link between both objects which is either an inter- or an intra-model link. *Unassociate* is the pending decision to *delete* an existing expansion link.

The mutation decisions affect what we call the *Temporal_environment* of an Object. This environment allows traceability from users' requirements to implementation. The purpose of a mutation link is to relate both input and output objects of a mutation. We have identify two types of mutation decisions : *mutate_to* (which creates the link) and unmutate_to (which removes it). A creation results either from a mapping decision (an inter-model mutation) or from a retyping decision (an intra-model mutation).

### 3.2 Guidelines for defining the way-of-working

We show here how to use the pre-defined set of generic decisions, based on our classification of object evolution, to define a way-of-working as a set of macro and micro contexts. The approach is based on the four following steps :

(1) Identification of the classes of objects of the RE methodology,
(2) Definition of the three environments for each object class,
(3) Definition of decisions by instanciation of the generic hierarchy (shown in figure 5) for each object class,
(4) Refinement of macro decisions for each object class.

We exemplify the four steps with part of the F3 methodology which maps the Enterprise model [1] into the Information System (IS) model [19].

Step (1) : The IS model is an E/R like model which also describes the services provided by an entity-type. We show, in figure 6, some classes of objects such as Service or Entity-type which correspond to the concepts of same name in the IS model. We look upon a concept as a class of objects in order to be able to describe the possible evolutions of its instances i.e. their possible transformations, expansions and mutations.

For instance, we have described the association of entity-types with two objects Role and Relationship at least for three reasons : (1) a Role may have several transformations (in figure 6, we can see that the object Role has three properties : role_name, cardm and card_max), (2) a Relationship may be expanded with attributes (see, in figure 6, the expansion link between Relationship and Attribute), (3) a relationship may mutate to an Entity-type (see, in figure 7, the mutation link between Relationship and entity-type).

The Enterprise model describes the environment of Information systems. In the F3 methodology we use it to elicitate users' requirements. In this approach, an analysis of the activities related to a future information system is generally a pre-requisite to its development. In figure 7, we show the main objects which are related to activity : Actor, Event, Resource and Goal.

Step (2) : In this step, we have to define for each class of object the three environments : inner, spatial and temporal. These environments may be shown graphically as suggested in figures 6 and 7 i.e. the nodes are object classes (normal boxes) or property classes (boxes with a black corner like in the meta model), the links have three different representation according to the emphasis

The two first environments are easy to build considering the semantic of each concept. In figure 6, we can see for instance that an Entity_type may have : (a) a name in its inner_environment, (b) a key, attributes, roles and services in its spatial_environment. The inner_environments related to the objects of the Enterprise model are not shown in figure 7 for the sake of clarity. The simplified description of the spatial environment of Activity is shown in the same figure : Event, Actor, Goal and Resource.

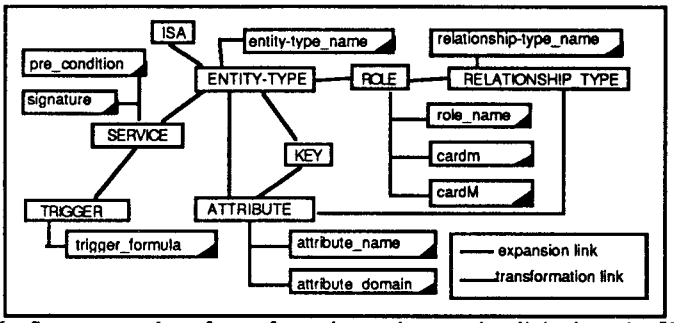


**Figure 6 :** Some examples of transformation and expansion links into the IS model

The temporal_environments buildings are more difficult to achieve. A method engineer must consider the semantic of concepts and the heuristics used by developers. Generally the mutations resulting from mappings or retypings relates objects, the spatial_environments of which have similarities i.e. in figure 7, we can see that an entity-type and a relationship may be both expanded with attributes. As the spatial_environment of a relationship is potentially more restricted than an entity-type one, we have defined a mutation link from relationship to entity-type and not the contrary. Of course, we could have both. The similarity between Activity and Service is more complex, yet it is easy to understand that they cover similar aspects with different languages.

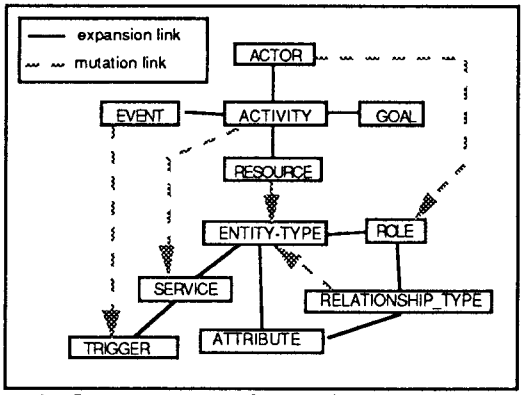


**Figure 7 :** Some examples of expansion and mutation links

Step (3) : In this step, the terminal leaves of the generic hierarchy are instanciated first (3.1) for each object to define a set of micro contexts i.e. the generic decisions are replaced by as many specific decisions as there are object classes and links. Then the non terminal leaves are instanciated (3.2) in turn for each object to define a set of macro contexts.

(3.1) To get micro contexts we must define : transformation decisions, expansion decisions and mutation decisions. A method engineer must have in mind that he should reuse as far as possible the decision names the Requirements engineers are familiar with. In the paper, we have tried to use decisions names which are easy to understand.

Figure 8 shows some of the transformation decisions (for the sake of clarity we show only the create and define decisions) related to the links mentioned in figure 6 : for each class there are a create and a delete decision ; for each transformation link there are a define, a redefine and an undefine decision. These decisions allow to define micro contexts related to object classes. A context rely a decision to the object(s) (i.e. the situation) on which it is

applied. For instance for the entity-type object class, we have two micro contexts (with a transformation intention) :

< ( ) , Create_Entity_type > / to create an object of this class
< ( Entity_type ) , Define_Entity_type_name > / to create a property name

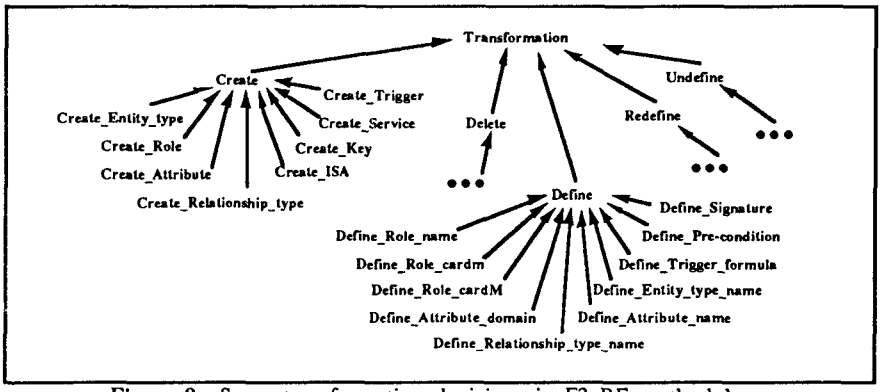


**Figure 8 :** Some transformation decisions in F3 RE methodology

Figure 9 shows the expansions decisions related to the links mentioned in figures 6 and 7. Specifically to each expansion link, there are an associate decision and an unassociate decision. For instance, figure 6 shows an expansion link between Entity-type and Attribute, therefore we can define two decisions : *associate_Attribute_to_Entity_type* and *unassociate_Attribute_to_Entity_type*. These decisions allow to define micro contexts too. For instance for the entity-type object class, we may define (with an expansion intention):

< (Entity_type, Attribute) , Associate_Attribute_to_Entity_type >
< (Entity_type, Role) , Associate_Role_to_Entity_type >
< (Entity_type, Service) , Associate_Service_to_Entity_type >

Figure 10 shows the mutations decisions related to the links mentioned in figure 7. Intra-model mutation links correspond to possible retypings i.e. either to improve or to correct a representation. The decision *Mutate_Relationship_to_Entity_type* belongs to this class. Inter-model mutation links, such as *Mutate_Actor_to_Role* or *Mutate_Activity_to_Service*, represent mappings from one level of modelling to another one.

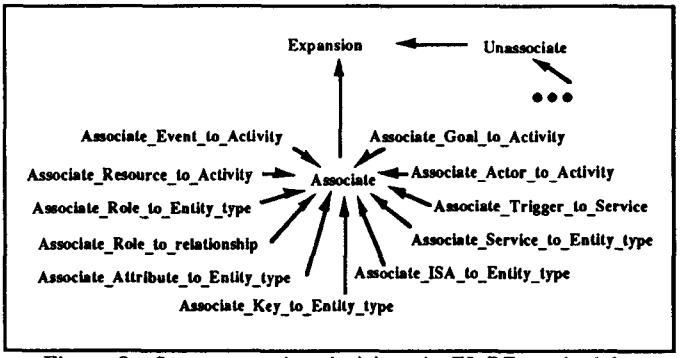


**Figure 9 :** Some expansion decisions in F3 RE methodology

These decisions allow to define the last sub-set of micro contexts. For instance for the relationship object class, we may define (with a mutation intention), the following micro context : < (Relationship) , Mutate_Relationship_to_Entity_type >

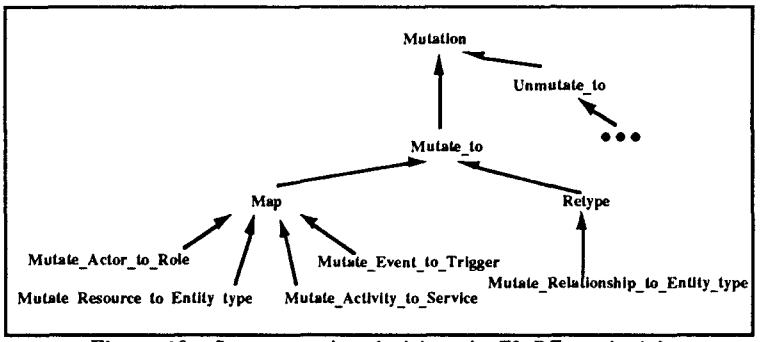


**Figure 10** : Some mutation decisions in F3 RE methodology

(3.2) To get macro contexts, we take into account the non-terminal leaves of the generic decision hierarchy (see figure 5) giving new names to them in order to define specific decisions.
For instance, related to Entity-type object class, we may obtain the following hierarchy of macro contexts (which is also completed with the micro contexts defined in step (3.1) :

< (Entity_type) , Evolve_ET > / Similar to the improve decision of figure 4.
- Alt1 < (Entity_type) , Transform_ET > / alternatives shown in figure 8
  - Alt1.1 < (Entity_type) , Define_Entity_type_name >
- Alt2 < (Entity_type) , Expand_ET > / some of the alternatives shown in figure 9
  - Alt2.1 < (Entity_type, Attribute) , Associate_Attribute_to_Entity_type >
  - Alt2.2 < (Entity_type, Role) , Associate_Role_to_Entity_type >
  - Alt2.3 < (Entity_type, Service) , Associate_Service_to_Entity_type >
- Alt3 < (Entity_type) , Mutate_ET > / no alternatives shown in figure 10

In fact, the number of micro decisions and by the way the richness obtained in the F3 process model is more important than the one sketched in these lines, because we should consider that association of concepts are objects too. For instance, in the Enterprise Model (EM) all the relationships are directional links which may also carry properties such as names. As a matter of fact, we may improve the part of the way-of-working dealing with transformations and mainly with mutations. For instance, if Actor may mutate to Role, Resource to Entity-type and there is an EM_Link defined between Actor and Resource, then we could define a mutation_link between this object EM_Link_Resource_to_Role and the object Relationship_type. At the further stage of process guidance, if the two first mutations have been performed, the guidance tool can suggest to perform the third one.

Step (4) : In order to take into account more specific aspects of the RE methodologies, we propose to extend the set of contexts systematically defined in two ways : (4.1) clustering decisions, (4.2) explicating specific mode of reasoning (such as a Top-down approach).

(4.1) Clustering of decisions is resulting from the possibility to define meaningful groupings of decisions. For instance, the F3 E/R like model has static aspects and dynamics ones. It is worth to group decisions according to this criterion when the intention is to refine an existing object. This introduces two levels more in the hierarchy as showed in figure 11. For instance, if we consider the alternatives related to Expand_ET (define in step 3.2) we can separate those which are dealing with the static definition of an entity-type from those dealing with its dynamic definition :

< (Entity_type) , Expand_ET >
  < (Entity_type) , Refine_ET_definition >
    < (Entity_type) , Refine_ET_Dynamic_definition >
      < (Entity_type) , Refine_ET_with_Service_definition >
    < (Entity_type) , Refine_ET_Static_definition >
      < (Entity_type) , Refine_ET_with_Role_definition >
      < (Entity_type) , Refine_ET_with_Attribute_definition >
      < (Entity_type) , Refine_ET_with_Key_definition >
      < (Entity_type) , Refine_ET_with_Sub-type_definition >
      < (Entity_type) , Refine_ET_with_Attribute_definition >

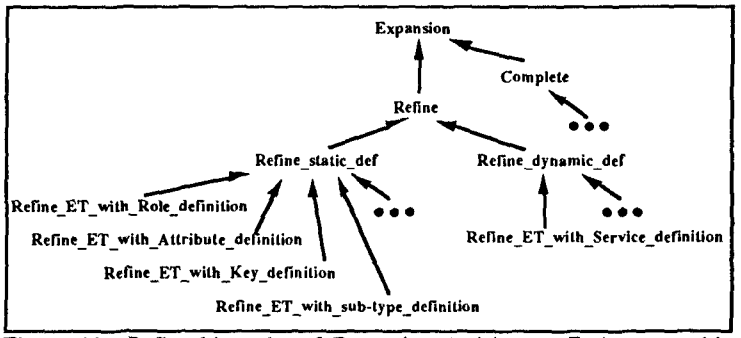


**Figure 11** : Refine hierarchy of Expansion decision on Entity_type object

(4.2) Traditionally, a methodology suggests either a Top-down approach, a Bottom-up approach or mix of both. This is what we call the mode of reasoning. The point is that a mode can influence the definition of the macro contexts. Let's consider the Refine_ET_with_Attribute_definition decision (similar to the decision called *Attributise* in figure 4) mentioned in previous example. The reasoning mode used to define an Attribute can be, as it is in the F3 way-of-working, the following :

< (Entity_type) , Refine_ET_with_Attribute_definition >
Alt1  < (Entity_type) , Define_new_Attribute_of_ET >
  Alt1.1< (Entity_type) , Create_new_Attribute_of_ET >
    < ( ) , Create_Attribute_by_mutation >
      < ( ) , Create_Attribute_by_mapping >
      < ( ) , Create_Attribute_by_retyping >
    < ( ) , Create_Attribute_by_reuse >
    < ( ) , Create_Attribute_from_scratch >
      < ( ) , Create_Attribute>
      < (Entity_type, Attribute) , Associate_Attribute_to_Entity_type >
  Alt1.2< (Entity_type) , Identify_in_Current_Product_existing_Attribute_of_ET >
    < (Entity_type, Attribute) , Associate_Attribute_to_Entity_type >
Alt2  < (Entity_type) , Refine_existing_Attribute_of_ET >
  Alt2.1  < (Attribute) , Refine_Attribute_static_definition >

This last decision is similar to the Refine_ET_static_definition decision defined previously. As a matter of fact we have illustrated a Top-down approach [11] i.e. starting from the intention to Refine an Entity-type static definition we have decided to refine the static definition of one of its attributes. We can also define the reverse reasoning to get a bottom-up approach. By the way, a method engineer has a complete freedom to define the reasoning-mode he wants to allow in a given way-of-working. Let's illustrate such a bottom-up approach with the same example.

```
< (Attribute) , Refine_Attribute_Static_definition >
    < (Attribute) , Involve_Attribute_in_ET_definition >
    Alt1    < (Attribute) , Aggregate_Attribute_to_new_ET >
        Alt1.1< (Entity_type) , Create_new_ET >
                < ( ) , Create_ET_by_mutation >
                    < ( ) , Create_ET_by_mapping >
                        < ( ) , Create_ET>
                        < (Resource, Entity_type) , Mutate_Resource_to_Entity_type >
                    < ( ) , Create_ET_by_retyping >
                        < ( ) , Create_ET>
                        < (Relationship, Entity_type) , Mutate_Relationship_to_ET >
                < ( ) , Create_ET_by_reuse >
                < ( ) , Create_ET_from_scratch >
                    < ( ) , Create_ET>
                    < (Entity_type, Attribute) , Associate_Attribute_to_Entity_type >
        Alt1.2    < (Attribute) , Identify_existing_ET >
                < (Entity_type, Attribute) , Associate_Attribute_to_Entity_type >
    Alt2    < (Attribute) , Refine_existing_ET_of_Attribute >
        Alt2.1    < (Entity_type) , Refine_ET_static_definition >
```

## 4 Conclusion

This paper is centred on ways-of-working definition. Our proposition is based on three process modelling abstraction levels which more generally allow three complete forms of support : (1) process trace, (2) way-of-working definition and (3) process guidance. We have detailed in the paper point (2) by providing a set of guidelines for method engineers. We show in figure 12, that we can structure the process knowledge in a repository. A process meta model and some guidelines are used to define ways-of-working (as described in this paper). Then the resulting way-of-working (or process model) is used manually or not for different processes by developers. Learning from traces allow to improve the way-of-working and so the guidance support by capitalising development heuristics which are numerous in Requirement Engineering.

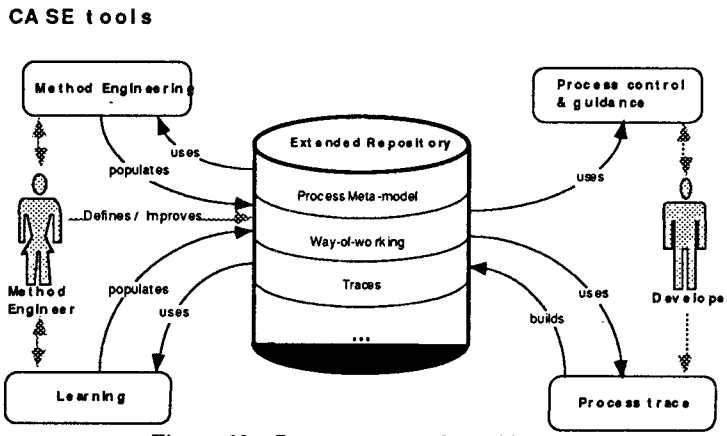


**Figure 12 :** Process case tools architecture

From the point of view of users, the Method Engineer knows and uses with a method engineering tool the process meta model to populate a way-of-working. Then using a learning tool, he is able to refine and complete the way-of-working by capturing Developers heuristics. The Developer uses the way-of-working definition with a process

control and guidance tool or manually if he wants to apply some new decisions which could be further capitalised. He also builds the trace as the process proceeds.
This architecture is developed within two Esprit III projects in the Requirements Engineering field. The F3 project aims at tracing the process. Its guidance tool will be based on a learning facility working on RE traces. These traces relate the objects with the three kind of links introduced here (transformation, expansion, mutation links) and also keep track of objects history in terms of all the decisions which have affected its definition. Finally, the method engineering approach defined in the paper is developed in this project.

## References


1. Bubenko J., Rolland C., Loucopoulos P., De Antonnellis V. : "Facilitating "Fuzzy to Formal" Requirements Modelling", Proc. of ICRE94, Colorado Springs, 1994.
2. Boehm B.W. : "A Spiral Model of Software Development"; IEEE Computer 21.
3. Curtis B., Kasner H., Iscoe N. : "A field study of the software design process for large systems"; Comm. ACM, vol. 31, 1988.
4. Dowson M. : "Iteration in the Software Process"; Proc 9th Int Conf on "Software Engineering", Monterey, CA,1988.
5. Finkelstein A., Kramer J., Goedicke M.: "ViewPoint Oriented Software Development"; Proc. Conf "Le Génie Logiciel et ses Applications", Toulouse,1990.
6. Grosz G., Rolland C. : "Using Artificial Intelligence Techniques to Formalize the Information Systems Design Process"; Proc. Int Conf "Databases and Expert Systems Applications", 1990.
7. Henderson-Sellers B., Edwards J.M.; "The Object-oriented Systems Life-Cycle"; Comm. ACM, Vol. 09, 1990.
8. Jarke M., Mylopoulos J., Schmidt J.W., Vassiliou Y.; "DAIDA - An Environment for Evolving Information Systems"; ACM Trans. on Information Systems, Vol. 10, No. 1, 1992.
9. Loucopoulos P., Zicari R.: "Conceptual Modeling, Database & Case", Wiley, 1992.
10. Moreno M., Souveyet C. : "The Evolutionary Object Model", IFIP TC8 Int. Conf. on "Information System Development Process", North Holland (pub.), 1993
11. Olle T.W., Hagelstein J., MacDonald I., Rolland C., Van Assche F., Verrijn Stuart A. : "Information Systems Methodologies : A Framework for Understanding", Addison Wesley, 1988.
12. Peugeot C., Franckson M.: "Specification of the Object and Process Modeling Language", ESF Report n° D122-OPML-1.0, 1991.
13. Potts C.: "A Generic Model for Representing Design Methods"; Proceed. 11th International Conference on Software Engineering, 1989.
14. Rolland C., Cauvet C. : "ALECSI : An Expert System for Requirements Engineering", in "Advanced Information Systems Engineering", Springer Verlag, 1991.
15. Rolland C.: "Modeling the Requirements Engineering Process", Proc. Fino-Japanese Seminar on "Conceptual Modeling", 1993.
16. Rolland C.: "Modelling the Evolution of Artifacts", Proceed. of Requirement Engineering Conference ICRE94, Colorado Springs, 1994.
17. Royce W.W.: "Managing the Development of Large Software Systems"; Proc. IEEE WESCON 08/1970.
18. Schmitt J.R.: "Product Modeling in Requirements Engineering Process Modeling", IFIP TC8 Int. Conf. on "Information Systems Development Process", North Holland (pub.), 1993
19. Sportes D. : "The Information System Model"; F3 Deliverable on T2.2, EspritIII project, n°6612, september 93.
20. Wilenski; "Planning and Understanding", Addison Wesley, 1983