Modeling the Requirements Engineering Process

Colette Rolland
Universite de Paris 1 Pantheon-Sorbonne UFR06
17, Rue de la Sorbonne   75231 Paris Cedex 05 FRANCE
email : rolland@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 modeling. This has caused the introduction of a large variety of models and especially conceptual models by which an information system can be modelled in high level conceptual terms. By contrast, very little attention has been paid to the conceptual modeling process which has the purpose of investigating the requirements of the users community and abstracting from that the conceptual specification of the information system. This results in a low level of support provided to requirements engineers. However, the emphasis on system modeling is shifting to process modeling. The need for process modeling motivates the process stream of the NATURE Esprit project. The Requirements Engineering Process model developed within this project is the main topic of this paper. The particular RE process modeling approach chosen in NATURE emphasises the notion of decision within the context in which it is taken. The paper outlines the modeling approach. It details and exemplifies the main concepts proposed to model the RE process and their relationships. Finally, it sketches the advantages of the process model by introducing its different usages.

1. Introduction

Information System Engineering has made the assumption that an Information System (IS) is supposed to capture some excerpt of world history and hence has concentrated on modeling, that is capturing information about the world.

It is usual to view an IS as "a model of some slice of reality of an organisation" [1] and even to regard the IS development as a problem of models construction and description. As shown in figure 1 these models are developed as part of two major development activities namely the Requirements Engineering and the Design Engineering activities.

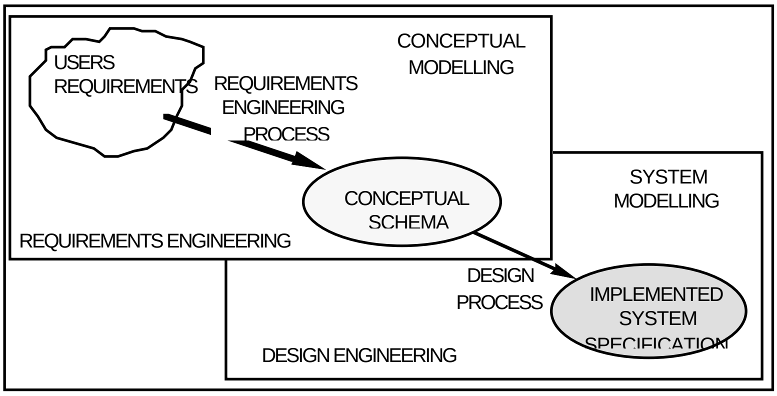


Figure 1 : IS development process

The term Requirements Engineering has been introduced by J. Hagelstein [2] and E. Dubois [3] for this part of the IS development that involves investigating problems and requirements of the users community and developing a specification of the future system, the so-called conceptual schema.

The succeding development part, where the specification is used to design and implement a working system that is specified against the specification, may be called Design Engineering.

This practise provides an answer of sorts to the fundamental question : what does the information handled by my 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 my information system?

The emphasis on *product* i.e. the system models has hidden the importance of *process* i.e. the route to deliver the product.

Our central thesis is that process modeling is as important as system modeling is. For lack of process modeling, understanding of what is the development process, what happens, when, why, on what, by whom, is very poor. The process semantics is not well captured, with the required level of detail, in existing process models.

Consequently the way-of-working prescribed by methodologies are hardly defined. They give few insights on how developers can/must proceed to progressively transform the initial requirements into an efficient and reliable computerised system which matches these requirements. Subsequently CASE tools are not able to provide a true assistance to developers. They are efficient in recording, retrieving and manipulating system models but are almost unable to actually support the developers in performing the creative activities of models construction and transformation.

What actually happens when developing an IS is not recorded. Tracing in a structured manner the history of an IS development, keeping track of each transformation of the product, of what occurred and when is almost not possible without an adequate process model. Moreover capturing process knowledge is important to reason about the consequences of changing conditions and requirements in system analysis, design and maintenance and to reuse design decisions in subsequent projects. Recent research works support this claim (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). In the NATURE Esprit project the need for process modeling motivates the "Process Theory" stream .

NATURE aims at investigating three aspects of the Requirements Engineering (RE) phase of IS development : (1) knowledge representation, (2) domain knowledge definition and usage, (3) RE process modeling.

The particular RE process modeling approach chosen in NATURE assumes that the basic building block of any process can be modelled as a 4-tuple : <*situation,decision,argument,action*> [4]. It associates the situation the requirements engineer has to deal with to one of the decisions he can take to solve the local problem, the argument that motivates the decision and one of the actions to be performed to apply the decision.

The paper is centred on the presentation of the key concepts proposed to model a RE process. This is made in section 4. In section 2, some basic definitions are provided. The state of art is outlined in section 3. The various usages of the process meta-model are finally outlined in section 5.

2. Some basic definitions

The term *process* is heavily used in various contexts. In the paper it has the following meaning "a related set of activities conducted to the specific purpose of product definition".

A *process model* is a description of process. Most often a process model is used for a descriptive purpose i.e. to describe "how things must/should/could/be done" in contrast to the process itself which is really what happens.

A process is then an instanciation of a process model which is executed. It corresponds to a development run whose output is a product, say a conceptual schema or a logical schema or the implemented information system.

In IS development, a process model corresponds to the way of working prescribed by the methodology in use. It is similar to the concept of plan.

The knowledge required to design such plans may be related to the third level of abstraction in process modeling (see figure 2) and may take the form of a *process meta-model*.

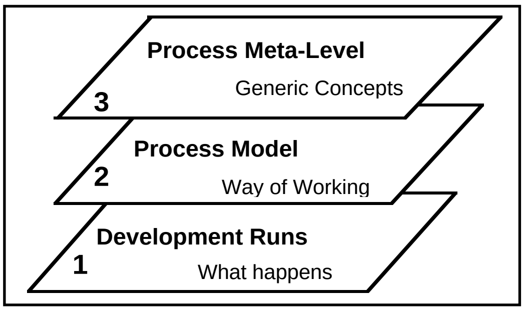


Figure 2 : Abstraction levels in process modeling

A process model is then an instanciation of a process meta-model. The latter plays a role similar to the theory of plans [5] from which plans can be generated (the process models) and executed (the processes).

The paper aims at presenting a process meta-model which has been tailored for the requirements engineering phase. It is being developed in the NATURE project and will be referred to as the NATPROC meta-model.

3. State of the art

According to the Dowson's [6] classification of existing process models into :

- activity-oriented models,
- product-oriented models,
- decision-oriented models.

The NATPROC meta-model falls into the third class.

Activity-oriented models come from an analogy with problem solving, i.e. finding and executing a plan of actions leading to the solution. They are sequential in nature and provide a frame for manual management of projects developed in a linear fashion. Such a linear view of the design process is inadequate for methodologies which support backtracking, reuse of previous designs and have to support parallel engineering activities. The first widely used model, the Waterfall model [7], falls into this category, along with the Spiral model [8] and the Fountain model [9] which try to eliminate the well recognised lack of flexibility of the Waterfall model. Limitations and drawbacks of activity-centred development approaches come from their representation of development processes like programs which do not integrate at all the interactive aspects of IS development.

The product-oriented process models represent the development process through the evolution of the product. They promote a view of development processes which is still centred around the notion of development activity but present the advantage to link development activities to their output : the product. The ViewPoints model [10] belongs to this category as well as the development process model proposed in the European Software Factory (ESF) project [11].

A more recent class of process models follows a *decision-oriented paradigm*. The successive transformations of the product are looked upon as consequences of decisions. The process model of the DAIDA project [12, 13] and of [14] fall into this category. Such models are semantically more powerful than the previous ones because they explain not only how the process proceeds but also why transformations happen. They provide a more complete knowledge about the process which is particularly useful for reuse of developed products, or of design decisions, and for backtracking purposes when a modification in requirements has to be considered either during the design process itself or during the maintenance of the IS.

Considering the highly non-deterministic nature of the RE process we do believe that only the last of these approaches appears to be partially appropriate. It is probably difficult to impossible to write down a realistic state-transition diagram that adequately describes what has to happen or actually happens in requirements engineering. But relying on the pure artefact history is also insufficient. Even the decision-based approaches offer only limited hints about when and how to decide on what.

Analysts react *contextually* according to the domain knowledge they acquire and react by analogy with previous situations they have been involved in. In order to take into account the very nature of the RE we have chosen to emphasise the contextual aspect of decisions. This approach strongly couples the context of a decision to the decision itself, otherwise we loose information about the decision. The NATURE 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).

4. The process model

The central aspect of the NATURE process way of modeling is therefore that it makes the notion of *situation* (in which to decide) explicit and relates it to the broader question of context handling.

Figure 3 gives a simplified overview of the process meta-model, introducing its key-concepts and their relationships (with a binary E/R based notation).This approach is an extension of the one presented in [4] and implemented in the ALECSI prototype [15].

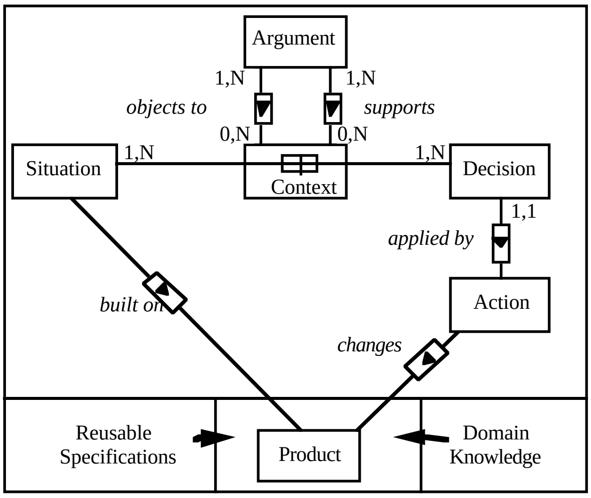


Figure 3 : Simplified overview of the Process Meta-Model

The four basic concepts are the ones of situation, decision, action and argument :

- *situation* to explain the decision context,
- *decision* to guide the RE process,
- *action* to perform the product transformations,
- *argument* to support decision-making.

A *situation* is most often a part of the product under development it makes sense to take a decision on. It represents the context of a decision. It can be a statement about the application domain or a requirement. It can also refer to part of existing products resulting from previous developments or a piece of generic domain knowledge.

A *decision* reflects a choice that a requirement engineer makes at a given point in time of the RE process. A decision can be looked upon as the intention of product transformations. It explicitly represents the resolution of an issue.

An *action* performs a transformation on the product. It is a materialisation of the decision. Performing it changes the product and may generate new situations, subjects to new decisions.

Arguments are statements which support or object to decisions within a given context.

NATPROC emphasises the dimension of decisions and introduces as a main concept of process modelling the notion of context. A context is a couple <situation-decision> which amplifies the decision semantics by refining the "when" part of the decision.

Furthermore, the Process meta-model has to handle different levels of granularity of contexts, i.e. granularity of situations, as well as of decisions and actions. Granularity is a key point for mastering the complexity of Requirements Engineering.

The refinement of the process model to cope with this granularity problem leads to the more detailed overview of the Process meta-model presented in figure 4.

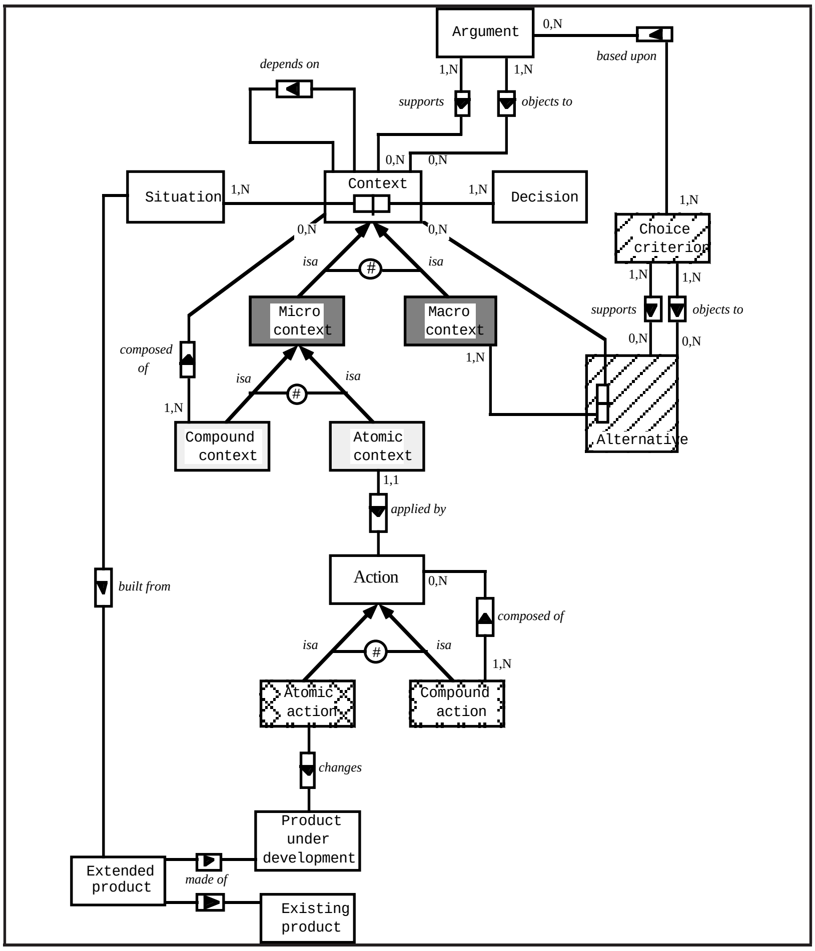


Figure 4 : Overview of the Process Model

It introduces specialisation of *contexts* into *macro* and *micro* ones (grey boxes in figure 4). The former corresponds to a situation which requires for its treatment to explore alternatives in decision-making. The latter refers to a context for which it exists only one resolution path.

A *macro-decision* refers to a class of decisions i.e. a set of decisions which correspond to the same intention of action. All the decisions of the set deal with the same issue but in different ways. Instances of the class correspond to various decision-making paths.

Figure 5 gives two examples of macro-decisions associated to the same situation (a sub-part of an Entity/Relationship schema under development for a subscription library system).

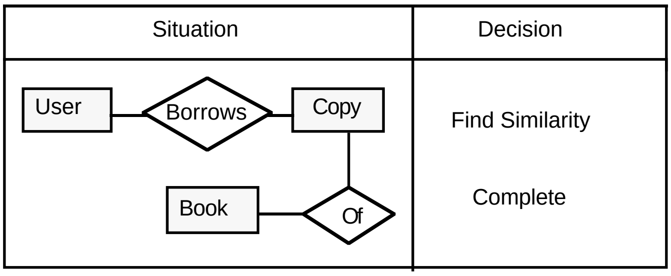


Figure 5 : Examples of macro-decisions

Similarities can be found in different ways : in NATURE for example two main approaches are experienced : one is based on measures of object distances [16] ; another one relies upon domain classification and characterisation [12].

Similarly the completion of the current product may be performed by adding attributes to entity types or relationship types or defining cardinalities on roles.

Alternatives of a macro-context are not necessarily exclusive and the various possible decision paths that can be performed are not necessarily ordered. In order to support the deliberation process to help decision making, the meta-model provides arguments. *Arguments* support or object to decisions within a context. "Reuse of modeling frames" will, for instance, support the "Find similarity" decision whereas " Not yet identified reuse criteria" will object to.

A choice criterion combines arguments in order to help the Requirement engineer in selecting the more appropriate alternative (hatched boxes in figure 4). In the case in hand, the choice criterion will suggest to make the "Find similarity" decision prior to the "Complete" decision in order to progress more efficiently in the schema definition. Reuse will avoid to perform work that has been already done in similar situations.

Decomposition of macro-contexts can be made iteratively. This is graphically represented in figure 4 by the loop upon context through macro-context and alternative. It means that one of the alternatives of a macro-context can be regarded as a macro-decision with several associated decision making paths. For this reason, one can speak of decisions upon decision. Such a decision is particular because its consequence is not an action on the product but the selection of one decision among a set of possible ones. The decision making process in this case, can be assimilated to the search of the right strategy for problem solving.

For instance, the two alternatives "Find similarity" and " Complete" previously discussed can be seen as the decomposition of the macro decision "Progress"; They are in turn decomposed as it is partially shown in figure 6 .

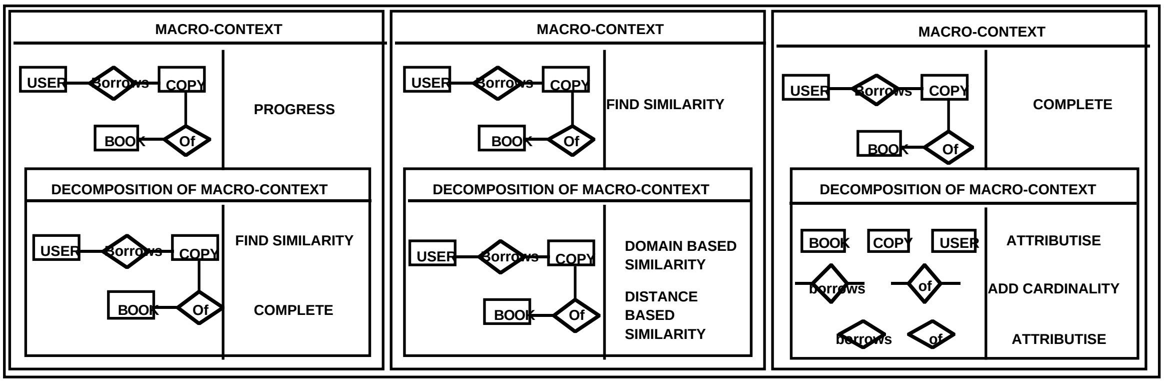


Figure 6 : decomposition of macro-contexts

This leads to the notion of a RE process guided by a hierarchy of decisions. Figure 7 is an attempt to classify the different types of decisions that a Requirements engineer could have to take during the RE process. The generalisation/specialisation abstraction mechanism is used to build the top level of the hierarchy of the basic kinds of RE decisions. All the mentioned decisions are generic i.e. independent of any methodology. The first level corresponds to the four main global strategies to tackle the Requirements Engineering problem (Freeze/Unfreeze, Progress, Backtrack, Check). The granularity of the decision decreases with the level of the node in the tree. For instance "Transform" is one of the way to "Progress"; "Improve" is a more specific mean to Progress through transformations of the product. "Add/Remove" product elements is one of the more specialised way to "Improve" the product proposed by our taxonomy of decisions.

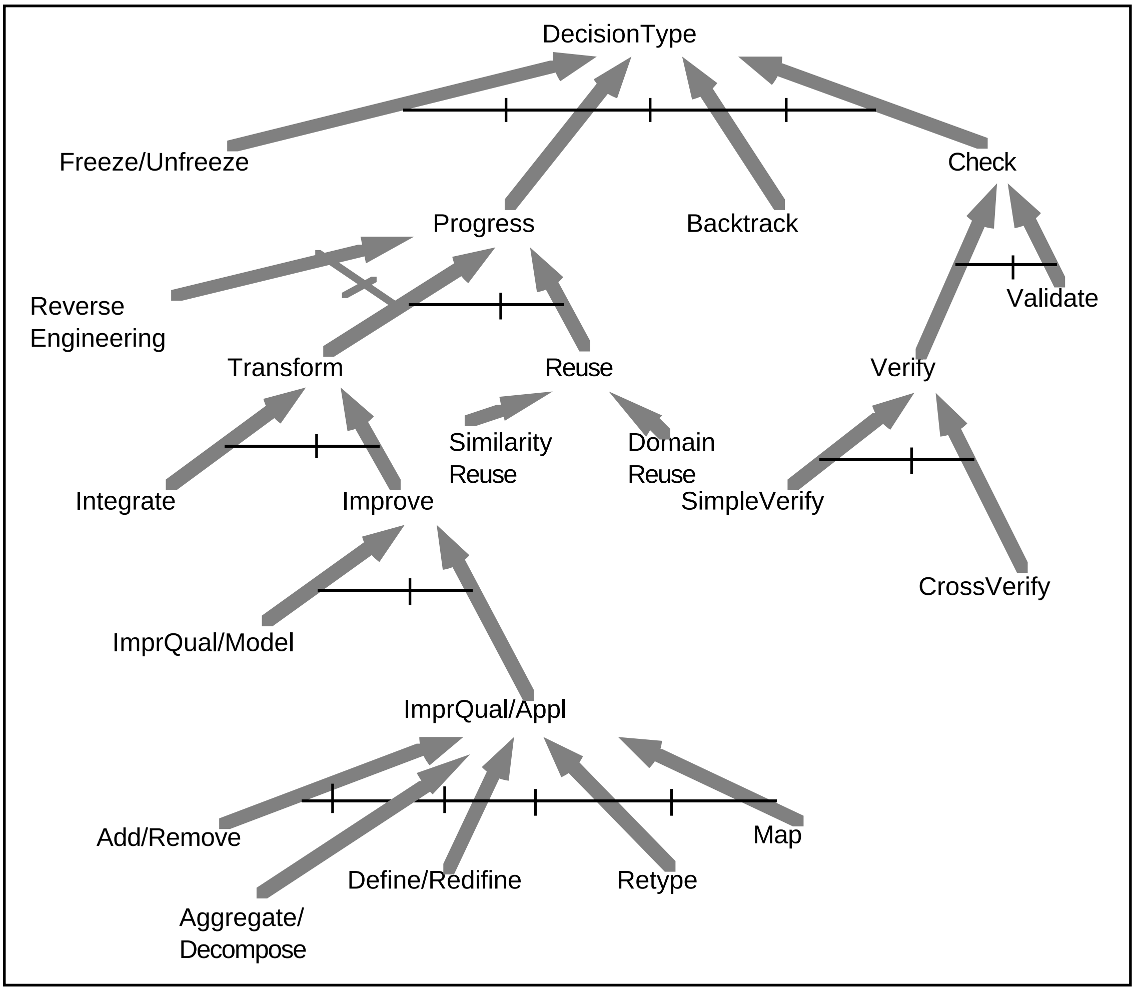


Figure 7 : Hierarchical taxonomy of decisions

When a context cannot be associated to multiple ways for solving the issue it raises, it is referred to as a micro-context.

A *micro-context* implements one decision making path. The three contexts of the figure 6c are examples of such micro-contexts. The "Partition" decision as part of the context stated in figure 8 is another example.

Consequences of micro-decisions are product transformations or refinements.

However, a micro-decision can itself be atomic or compound (dotted boxes in figure 4). An *atomic decision* triggers one action which can be in turn (cross-hatched boxes in figure 4) *compound* (composed of several actions) or *atomic* (changing only one product element).

A *compound-decision* is further decomposed in other decisions to be taken on sub-situations of its initial micro-context. This is modelled through the decomposition of contexts ("composed of" relationship in figure 4). For instance, the decomposition of the compound decision "Partition" is as follows :

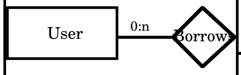
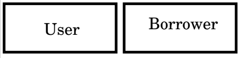
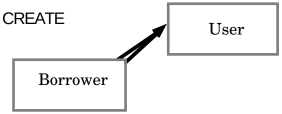
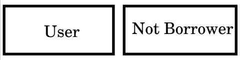
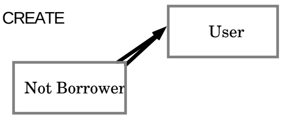
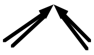
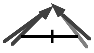
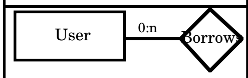
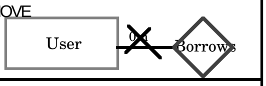
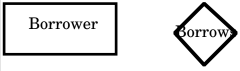
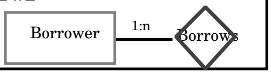
COUMPOUND CONTEXT	ATOMIC CONTEXTS		ATOMIC ACTIONS
	SITUATION	DECISION	
 PARTITION		BORROWER CREATION	CREATE
		BORROWER SUB-TYPE CREATION	CREATE 
		NOT BORROWER CREATION	CREATE Not Borrower
		NOT BORROWER SUB-TYPE CREATION	CREATE 
		DEFINE PARTITION CONSTRAINT BETWEEN BORROWER & NOT BORROWER	DEFINE 
		USER-BORROWS ROLE REMOVAL	REMOVE 
		BORROWER-BORROWS ROLE CREATION	CREATE 

Figure 8 : Example of compound decision

The initial micro-context <User/0.N/Borrows> is further decomposed into seven atomic contexts including each, a sub-micro situation of the initial one and the corresponding micro-decision. Each micro decision is implemented through the execution of one action which creates or removes elements of the product under development.

Micro-decisions are *methodology dependent.* Some of them relate to the concepts of the various models supported by the methodology. Others derive from the way-of-working prescribed by the methodology.

Let us call *decision unit* either a concept or a property of concept of a given model. Entity-Type, Attribute, Domain, Role, Relationship-Type are examples of concepts of the E/R model; Valuation is called a property of the Attribute concept (attributes can be mono-valued or multi-valued); Cardinality is a property of the Role concept and Key is a property of the Entity-Type concept. Our suggestion is to associate primitive decisions to each unit. Furthermore primitive decisions relating to concepts are of four basic types : Add/Remove, Retype, Map and Aggregate/Decompose; the ones derived from properties of concepts are only of the type Define/Redefine. For example it makes sense to Add/Remove an Entity-

Type, a Relationship-Type, an Attribute, a Domain or a Role. Define Key of an Entity-Type, Define Cardinality of a Role and Define Valuation of an Attribute are valuable decisions.

These primitive and methodology dependent decisions can take place in the hierarchical taxonomy of decisions outlined in figure 7. They correspond to sub-types of the leaves of the hierarchy. This is exemplified in figure 9 with primitive decisions associated to the E/R model.

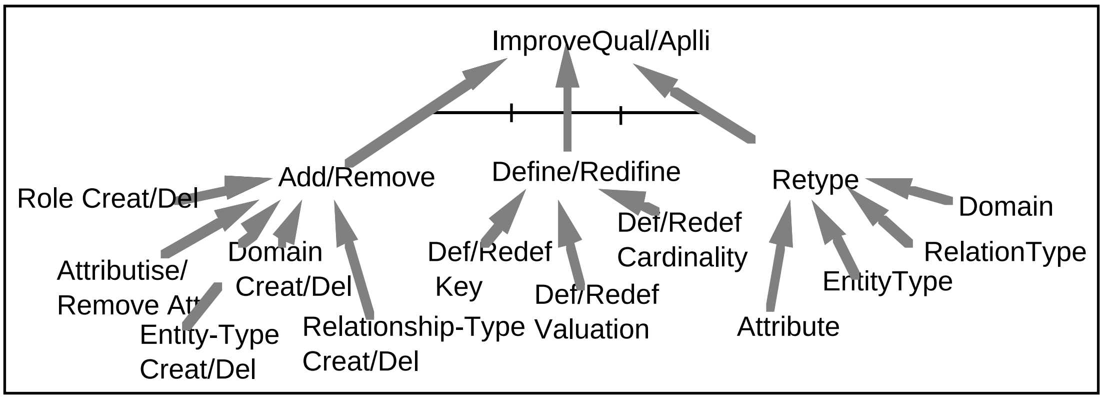


Figure 9 : Specialisation of the decision hierarchy for the E/R model

To summarise, we can say that granularity in decision-making relates to consequences of decisions. Some decisions have consequences that are only product transformations whereas others imply the taking of decisions as consequences. The former are referred to as macro-decisions and the latter as micro-decisions. Macro-decisions can be qualified as strategic ones (they refer to a strategic intention); micro-decisions implement a tactics.

The Requirements engineering process as modelled by NATPROC can been seen as composed of two interrelated loops : the strategic and the tactical loops (figure 10). The Requirements engineer logically starts entering the strategic loop and proceeds in the decision process till it reaches a level of detail which allow him to move to the tactical loop and to perform the consequences of the taken decision. He may then, proceed in a similar manner for another situation.

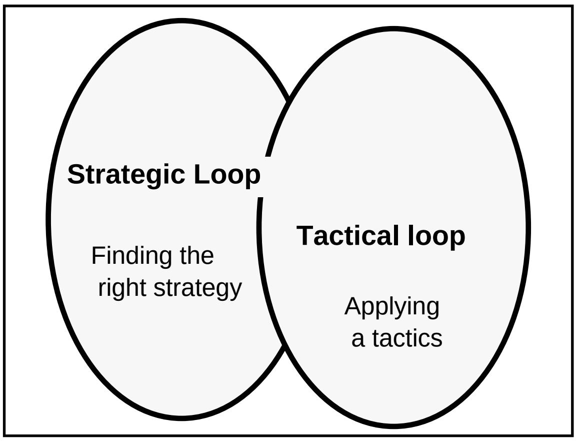


Figure 10 : The decision process

Moreover decisions are not always independent one to the others. *Dependency* among decisions is modelled through the "dependent on" relationship in Figure 4. For instance, the "Define key" decision is said to be dependent on the "Entity-Type Creation" decision; it means that the decision to define the key of an Entity-Type must be taken sometimes after its creation.

Expressing decision dependencies is a means to prescribe what has to be done consequently to what currently happens in the RE process. For example, linking the "Define Key" decision to the "Entity-Type Creation" decision will force the Requirements engineer to fulfil the Entity/Relationship model prescription which states that "any entity-type must have a key". The enforcement of decision dependence is a way to ensure the completeness of Requirements Engineering.

Dependencies are of two kinds : structural and definitional dependencies. *Structural dependencies* come from relationships existing among the concepts of the conceptual model in use. For instance, due to the definition of a relationship-type within the E/R model, the "Entity-Type Creation" decision belongs to the set of dependent decisions of the "Relationship-Type Creation" decision. *Definitional dependencies* relate to the definition of concepts and their associated constraints. In the E/R model attributes are defined as mono-valued or multi-valued; consequently the "Valuation" decision is dependent on the "Attributise" decision. Similarly, as cardinality constraints are associated to roles, the "Define Cardinality" decision is part of the set of dependent decisions of the "Role Creation" decision. Figure 11 illustrates sets of dependent decisions defined among the decisions outlined in figure 9 for the E/R modeling approach.

Primitive Decisions	Dependent decisions
Entity-Type Creation	{Define Key}
Role Creation	{Entity-Type Creation, Relationship-Type creation, Define Cardinality}
Attributise	{Entity-Type Creation v Relationship-Type Creation Domain Creation, Define Valuation}
Define Key	{Attributise}
Define Cardinality	{Role Creation}
Define Valuation	{Attributise}

Figure 11 : Dependent decisions sets

5. Using the process meta-model

From a theoretical point of view, the Process Meta-Model explains which are the key concepts needed to describe what happens in the development process, on what, when it happens and why.

From an operational point of view, the Process Meta-Model is aimed at providing guidance for method engineers and application developers.

The Process Model defines a set of generic concepts, i.e. *independent* of any RE methodology. It helps the method engineers in defining the way-of-working of the specific methodology they have to develop. The way-of-working of a given methodology will be obtained by instanciating the generic concepts of the Process Model. The result of such an instanciation consists of a collection of types, e.g. situation-types, decision-types or context-types. For example, the instanciation of the Process Model on an ER-oriented methodology would define the following context-type :< ER-schema , "Find similarity" > which states that the developer can take the "Find similarity" on the ER-schema under development.

NATPROC, the NATURE way-of-working is currently being defined and progressively refined as a way to experiment the approach and prove the theory.

The Process Meta-Model is also intended to help the application developers by providing the basis for an *automated guidance* of the RE process. Indeed, the definition of a RE methodology as instanciation of the Process Meta-Model can be viewed as a collection of process chunks that can be stored in a Process Knowledge Base of a Requirements Engineering CASE environment (figure 12).

Finally, instances of the collection of types defining the way-of-working of a specific methodology can be stored, in the Process Knowledge Base, to keep track of the RE process history. According to the contextual approach of the NATURE Process Model, the trace will bring a deep process knowledge allowing in turn backtracking and reuse. It will in addition give feedback to the method engineers and help them in improving the methodology way-of-working.

6. Conclusion

In this paper, we proposed to model the Requirements Engineering (RE) process as a network consisting of situations, decisions, arguments and actions. The relationships between situations, decisions, arguments and actions are denoted by the links between the nodes. The meta-model recognises the context in which a decision occurs, the need for argumentating the decision, and the outcome of the decision. It then, intimately links the process and the product by representing the RE artefacts as inputs and outputs of the process.

We argue that a process meta-model can support the various stakeholders involved in information systems projects by providing a means for capturing in a structured manner the history of a requirements engineering. It helps the method engineer in the definition of a way-of working which prescribes when things can/must be done in the development of information systems. It is required for the development of a knowledge-based software tool which intelligently assists the developer by checking the reasonableness of decisions, filling in missing details, providing advice, explaining actions and decisions and guiding the undergoing RE process.

Although our meta-model has been developed in the context of Requirements Engineering, we believe that its primitives are sufficiently generic to be applicable across the various phases of systems development. We are currently investigating the applicability of the model to all phases of the development of a large air-control system.

7. References

[1] : Loucopoulos P., Zicari R.(eds) : "Conceptual Modeling, Database and Case", Wiley, 1992.

[2] : Hagelstein J. : Declarative Approach to Information Systems Requiremnts", in Knowledge-Based Systems, Vol&, No4, 1988.

[3] : Dubois E.,Hagelstein J, Rifaut A. : "Formal Requirements Engineering with ERAE", Philips Journal Research, Vol L 43, No 4, 1989.

[4] : Grosz G. : " Using Artificial Intelligence Techniques to Formalize the Information System Design Process"; Proc Int Conf on Databases and Expert SystemsApplications, 1990.

[5] : Wilenski; "Planning and Understanding", Addison Wesley, 1983

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

[7] : Royce W.W.; "Managing the development of large Software Systems"; Proc. IEEE WESCON 08/1970.

[8] : Boehm B.W. : "A Spiral Model of Software Development and Enhancement"; IEEE Computer 21.

[9] : Henderson-Sellers B., Edwards J.M.; "The Object-oriented Systems Life-Cycle"; Comm. ACM, 09/1990.

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

[11] : Peugeot C., Franckson M.; "Specification of the Object and Process Modelling Language", ESF Report n° D122-OPML-1.0, 1991.

[12] : Jarke M., Mylopoulos J., Schmidt J.W., Vassiliou Y.; “DAIDA - An Environment for Evolving Information Systems”; ACM Transcactions on Information Systems, Vol. 10, No. 1, 1992.

[13] : Rose T., Jarke M., Gocek M., Maltzahn C., Nissen H. : " A Decision-Based Configuration Process Environment", Software Engineering Journal, 6, 5.

[14] : Potts C.; “A Generic Model for Representing Design Methods”; Proceed. 11th International Conference on Software Engineering, 1989.

[15] : Rolland C., Cauvet C. : "ALECSI : An Expert System for Requirements Engineering", in "Advanced Information Systems Engineering", R.Andersen, J.A.Bubenko, A.Solvberg (Eds), Springer Verlag, 1991.

[16] : Jarke M., Bubenko J, Rolland C., Sutcliffe A., Vassiliou Y.; "NATURE Initial Integration Report", NATURE.D-I, 1992.