AN APPROACH FOR DEFINING WAYS-OF-WORKING

Colette Rolland, Carine Souveyet, Mario Moreno

Université de Paris1 Panthéon-Sorbonne
CRI, 17 rue de la Sorbonne 75231 Paris Cedex 5, France
e-mail : {rolland,souveyet, moreno}@masi.ibp.fr

Abstract- Process modelling is considered today as a key issue by both, the Software Engineering (SE) and the Information Systems Engineering (ISE) community. However, most recent process models are descriptions of software development activities written in an executable form that computer systems can enforce. There is a need for process models which take into account heuristic knowledge to guide humans performing systems development. We refer to such guidance centred process models as 'ways-of-working'. So far, most work has concentrated on developing and experimenting with process modelling approaches and little attention has been paid to the problem of developing a method for defining a new process model. We propose in this paper, an approach for defining ways-of-working in a systematic manner. It is a meta-modelling approach in which a given way-of-working is constructed by instantiation of a process meta-model allowing to deal with a large variety of situations in a flexible, decision-oriented manner. Based on the properties of the meta-model, ways-of-working are formally defined, with an appropriate level of genericity, and in a modular way to facilitate their evolution and improvement. They are designed for providing automated and flexible guidance in decision making during the process. The paper presents the approach, exemplifies it with the way-of-working defined within the Esprit project F3 and illustrates how the process is guided on an F3 case study.

1 INTRODUCTION

Process modelling is considered today as a key issue by both the Software Engineering (SE) and the Information Systems Engineering (ISE) communities. Recent interest in process modelling is part of the shift of focus from the product to the process view of systems development. 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. Recent in-depth studies of software development practices [32], 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 for "a conceptual [process] model framework" [16].

Process modelling is a rather new research area. Consequently there is no consensus on what is a good formalism to represent processes or even on what the final objectives are [1]. Process models may be constructed for a number of different reasons, to fulfil different purposes. One purpose may be purely *descriptive,* that is, to record how some process or class of processes is actually performed. Alternatively, models may be constructed to guide, support and provide advises or instructions to developers, i.e.: to be *prescriptive*. The SE community has focused on descriptive models more than the ISE community.

Yet another way of looking at process models is in terms of the process aspect that they address : some focus on *managerial aspects* of the development process whereas others have *technical concerns.*

What is the paper about

In this paper, we are concerned with *modelling the technical aspects of the information systems development process in a prescriptive manner.* Defining guidance centred process models is the *first issue* we are dealing with. These process models are intended to form the basis of CASE environments and to provide these environments with the ability to guide application engineers in systems development.

There is a need for *guidance centred process models,* in particular, in the early phase of requirements engineering and conceptual modelling. While performing highly intellectual and creative activities needed to create the conceptual specification of the system, application engineers need to be guided and advised, locally to handle the particular situation they are faced to, and globally, to monitor the flow of decisions they have to make. The required support is based on heuristic knowledge, for example, suggestions on how to proceed in a certain context, providing alternatives for achieving a goal and advice to refine a goal. Neither CASE tools supporting current ISE methodologies nor SE centred environments are able to fulfil this need. CASE tools help in capturing, storing and documenting IS products but do not support application engineers in their creative development activities [34]. Process-centred software development environments are essentially enforcing the process performance that conforms to some preceptive software process definition [16].

The *second issue* in process modelling we are concerned with, is the process model definition. The paper advocates a *systematic approach for defining process models*.

Indeed, process performance based on a process model requires first, the capture of the process model. So far, most work has concentrated on developing and experimenting with process modelling approaches and little attention has been paid to the problem of developing a systematic method for defining a new process model [16]. Traditional information systems methodologies were defined in an "ad-hoc" manner. The process models underlying their development are expressions of the experiences of their developers. This has two major consequences : it is not possible to know how they were generated, and they become dependent on the domain of experience. Clearly, if process models are to be domain independent and if they are to be rapidly generatable and modifiable, then we need to go away from experience based process model definition. The ISE community has been recently paying attention to this problem adopting a method engineering approach [33], [48], [10] in which the process model is considered as the product of an engineering process. The SOCRATES project [23], [46] and the CAME (Computer Aided Methodology Engineering) approach [35] are early and promising works dealing with an automated support for creating situation specific process models.

We propose in this paper *a well-defined and repeatable approach to generate well formed guidance centred process models.* For guidance centred process models to be well-formed, we have identified a list of *six requirements.* This list takes into account the various shortcomings of methods which have been reported in the literature [41], [11], [18] and also, some of the criteria established by the software community to define a "good" process model [16].

Requirements for process centred process models

Granularity is the first important requirement for a process model. Most traditional process models (such as the waterfall model) are too large-grained descriptions of the product life-cycle. The granularity of the process steps is too large and descriptions are not detailed enough to provide an explicit guidance. The guidelines proposed to support the design of an Entity/Relationship [12] schema which consists of four steps :

-define entities
-when finished, define relationships
-when finished, assign attributes to entities
-when finished, attach further constraints

are a good illustration of this. Such a description is not accurate enough to provide guidance allowing, for instance, to choose a bottom-up or top-down design approach [3], or to decide if an entity should be retyped, decomposed or subtyped.
At the other end of the spectrum, there are models extremely detailed. For example specifying that after a source code file is edited, it should be recompiled [29]. Recently, hybrid formalisms that use different notations for large-grain and small-grain aspects of process such as PROCESS WEAVER [20] have been developed. We believe that a single process modelling formalism should accommodate a wide range of model granularity in a homogeneous fashion.

The *underlying paradigm* is the second characteristic. The activity-based paradigm is the prevalent view of the software process. It is a view that concentrates on the activities performed in producing a software

system and their ordering. This led to life-cycle descriptions most often treated as linear sequences, where crucial aspects of the process such as feedback loops and iteration are not represented [9], [13] and [14]. This emphasis incurs some risks of neglecting the influence of product structure on the development approach. For instance, the way the product is structured may have a strong influence on the steps needed for its construction. Furthermore, this linear view of process seems inadequate to model processes such as the requirements engineering process which are highly non deterministic in nature[28]. It is probably difficult, if not impossible, to write down a realistic state-transition diagram that adequately describes what has to happen during requirements engineering. The paradigm underlying a process model as to fit the process model purpose.

Genericity is another important requirement. Genericity provides the way for describing a general solution for a set of related problems, by parameterizing it with respect to its possible instantiations. Before being used, a generic solution has to be instantiated for a specific problem. There have, of course, been a number of attempts to define generic models (e.g. the spiral model [9] or the waterfall model[8]). However these attempts provide a very incomplete coverage of the set of development activities restricted to a broad description of the product life-cycle. In addition they were based on informal means of process definition, and this has made them difficult to follow systematically.

Self-improvement is our fourth recommended characteristic. Today, the process knowledge acquired during the development of a project cannot be easily capitalised to improve the current process model. Process models should help to materialise previous experiences in terms of knowledge which may be reused to guide processes in new projects. Presently, reuse of process knowledge is only performed manually.

Evolution support is the fifth requirement for process models. The production of systems is an evolutionary activity. It is clear that during this activity, not only do requirements and products change but also the process itself changes. In addition, it is possible for the process to change even while it is being executed. Accommodating this dynamic evolution of processes changes is a key but unresolved issue in process centred development environments. Process models should be designed to support this dynamic evolution of processes.

Guidance is the last characteristic of a process model. Both the ISE [36] and the SE [17] communities acknowledge the importance of process guidance. However despite thirty years of information systems development methodologies [36], [31] the level of guidance provided to application engineers by methods and CASE tools is still low, especially in the early phases of information systems development. Methods are often incompletely and unclearly defined whereas a formal and detailed description based on notations having a well defined semantics is required to provide automated guidance during process performance. Consequently, current CASE tools are essentially passive; they help in capturing, storing manipulating and documenting products but offer very little to support the largely intellectual process of product construction. Most process centred software environments [25],[4],[2],[29] provide active guidance [16] which consists of strict enforcement of a process model. In fact, as indicated by [45], process models are used to describe tools activity and to allow automatic invocation of tools. Process models are used to guide tools usage and not to support the application engineer in the performance of his/her intellectual activities. Sugiyama [43] nicely phrases the difference in referring to the latter as process scripts (the process models to support humans) whereas the former are called program scripts (the process models to guide tools). Our concern is process scripts.

the overall framework

The systematic approach being proposed in this paper is part of a larger approach to the problem of generating guidance centred process models which fulfil the fore-mentioned requirements. We will refer to these models as '*ways-of-working'* since they are intended to guide application engineers in their way of working to solve the problem in hand. This conforms to the view adopted in [7], [5], [42] and [46].

We propose a *meta-model* as a basis for process model definition. We understand a process meta-model to be a collection of concepts and inter-relationships between them. Since a process meta-model carries information about the process model, an instantiation of it shall result in a process model. Thus, our

main claim in this paper is the laying down of a rule, an approach, to process model definition : *the instantiation approach*.

Under the instantiation approach, the crucial issue in process modelling is no longer the process model but the process meta-model. After all, process models follow from a fairly straightforward instantiation of a process meta-model. This implies that the properties associated with the process meta-model get carried over to the instantiated model. Thus, if a process model is unable to handle different levels of granularity well, then this deficiency must be flowing from the meta-model. Similarly, if a process model is resistant to change, then the underlying meta-model must be hampering rapid change. This means that the onus of providing a process model with the required properties shifts from the process model developer to the process meta-model developer. If you have a 'good' meta-model then you can have a 'good' model and not otherwise.

This has the advantage that it forces us to find generic solutions to the problems of process models. Thus, we need generic solutions to granularity handling, handling evolving and self-improving process models and so on.

In this paper, we explore the i*nstantiation approach to the definition of guidance centred process models.* The approach is part of a more general CASE based framework for process definition, tracing, guidance, evolution and improvement which is illustrated in figure 1.

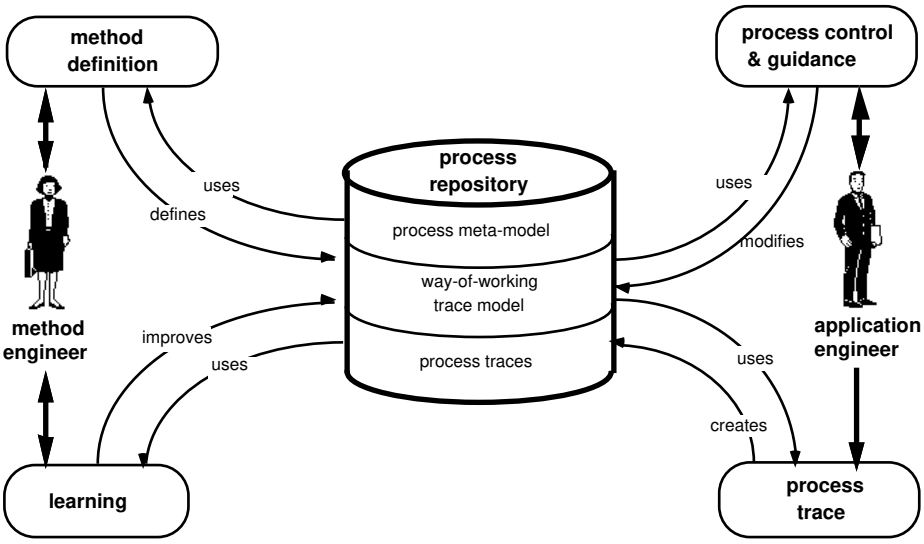

Figure 1 : The framework for process definition, trace and guidance

The first key point of the approach is to introduce three levels of process modelling reflected in the CASE repository.

-At the lowest level, process traces are recorded. A process trace records what actually happens during a development run, why it happens, when and on what it happens. There is one process trace for each process i.e. each application developed.

-At the second level, ways-of-working and trace models are defined. A process is then, an instantiation of a process model which is executed. A way-of-working is a process model [19] i.e. a description of process. It is intended to describe "how things must/should/could be done [21]. It has a prescriptive purpose and is similar to the concept of plan.

-The knowledge required to design such plans is related to the third level of abstraction and takes the form of a process meta-model. A process meta-model provides a set of generic concepts for describing ways-of-working which are therefore, instances of the process meta-model.

The latter plays a role similar to the Theory of Plans [47] from which Plans can be generated (the ways-of-working) and Executed (the processes).

There are also some similarities between our approach and the one advocated in Information Resource Dictionary Framework standard (IRDS) [24]. The IRD level, IRD definition level and IRD definition schema level are similar in nature to the three abstraction levels of our approach but they deal with levels of product description whereas ours deal with various levels of description related to the process of building products.

The second aspect of the proposal is to combine a top-down approach for way-of-working definition, customisation and process guidance to a bottom-up approach for way-of-working improvement and evolution.

- Assuming the existence of the process meta-model, the way-of-working corresponding to a given methodology will be defined by instantiation of the process meta-model concepts. This should avoid the definition of ways-of-working focusing on specific product engineering and failing to cope with many other development activities. The process model for a specific project will, in addition, be derived from the way-of-working by customisation of the way-of-working. Both, the way-of-working and its customisation are stored in the CASE repository. Thus, the process can be monitored and guided by the process model. Additionally, it can be traced according to the trace model.

- Even as the process proceeds, the process model can be adapted to specific requirements. and, at the end of the project, the way-of-working will be improved based on observation and analysis of process traces. This will allow the application engineer to benefit from past experience. Process knowledge, then, becomes independent of individuals and process reuse may become a company standard. Thus, instantiation and enacting may interleave like in the ALF approach [6].

The third contribution of the approach relates to the ability of the process meta-model to deal with many different situations in a flexible, decision-oriented manner. Moreover the meta-model can support different levels of granularity in decision making as well as non determinism by not insisting upon a specification of how a choice among open alternatives is actually performed. It identifies a decision in context as the basic building block of ways-of-working and allows their grouping into meaningful modules. Parallelism of decisions and ordering constraints are also supported.

To summarise, in the overall approach we adopted, we treat the three requirements of granularity, genericity, and guidance as integral to a meta-model : these are looked after in the concepts of the meta-model. However, the remaining three requirements, namely, paradigm, self improvement and evolvability, are part of the overall approach and the meta-model is built to support these features. It achieves this by relying on an extremely modular structure.

As outputs of our approach, ways-of-working are intended to be formally defined, with an appropriate level of genericity, in a modular way facilitating their evolution and improvement. As their underlying paradigm is decision and situation oriented, they are designed to provide an automated and flexible guidance in decision making, at different levels of granularity, while the process proceeds. The way they intend to fulfil the requirements for guidance centred process models is summarised in figure 2.

characteristics of process models	Our proposal
granularity	variable grain according to need
paradigm	situation and decision oriented
genericity	method independent
guidance	flexible choice-based
self-improvement	capitalising process experiences
evolution support	modularity

Figure 2 : Our position on criteria defined for "good" process models.

The approach has been developed as part of the process theory of the NATURE[1] project [27] and has been validated with the particular way-of-working of the F3[1] project [11]. We focus in this paper on the definitional process of ways-of-working which will be illustrated with the F3 methodology. Examples will be taken from an Air Traffic Control (ATC) case study performed within the F3 project.

The organisation of the paper is as follows. In section 2, we describe the process meta-model. In section 3, the method for defining ways-of-working is presented and exemplified with the F3 way-of-working. In section 4, we sketch how guidance is provided by the way-of-working. Finally, in section 5 we show how the requirements listed above are met and draw some conclusions.

2 THE PROCESS META-MODEL

2.1 The underlying paradigm

If we want a general frame to formally define ways-of-working for a wide range of methodologies, then, first of all, we need a process meta-model which provides a set of generic concepts allowing a proper description of what occurs in the development process, when it occurs, why and on what it happens. This will ensure the genericity feature of the ways-of-working generated from the meta-model.

In the literature, several process models have been proposed which have been classified in three major categories [15]: *activity-oriented* models, *product-oriented* models, and *decision-oriented* models We believe that decision-oriented models [26], [37], [30] are the most appropriate for guiding the development process. Such models are semantically more powerful than the prevalent activity-oriented models because they explain not only how the process proceeds but also why transformations happen. Therefore, they are more adapted to the true nature of the development process which consists mainly in making decisions of different kinds. Much of the requirements analysis phase involves discussions aimed at polishing an initial set of requirements into a precise one, and making decisions about what is to be modelled and how. The design phase might involve making choices among representations. During the maintenance phase decisions of changes have to be made. Guiding the development process thus requires to model decisions, alternatives in decision making, rationales of decisions and the deliberation process behind the decision process as well.

The process meta-model introduced in this paper is decision-oriented. Besides, we recognise the importance of the context in which decisions occur and in which arguments will support or object to decisions. The particular development process modelling approach we chose, emphasises the notion of decision within the context it is made. Application engineers 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 this basic nature of the development process 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. In short, our approach aims at capturing not only how activities are performed during the RE process but also why these activities are performed (the decisions) and when (the decision contexts).The underlying paradigm of the meta-model is a contextual approach.

The process meta-model was first sketched in [22] and refined in [39]. A first implementation has been done in the ALECSI prototype [38].

2.2 Presentation

2.2.1 The concept of context

Figure 3 presents a global view of the meta-model using some binary ER-like notations. A large box represents an entity-type and a small box represents a binary relationship between two entity-types. The arrow head indicates the direction in which the label of the relationship holds. For example, "Condition"

---

[1] This work is supported partly by the Basic Research Action NATURE (N° 5363) which stands for Novel Approaches to Theories Underlying Requirements Engineering and, partly by the ESPRIT project F3 (N° 6612) : "From Fuzzy to Formal".

and "Argument" are entity types and are related through "is based on" relationship. The direction of the relationship says, "Condition is based on Argument".

The model also includes the notion of an "objectified relationship" [44]. This notion is an abstraction mechanism which allows a relationship to be viewed, at a higher level of abstraction, as an entity-type. This applies for example, to the relationship "alternative" between a "Choice based Context" and a "Context" which is viewed as the entity-type 'alternative' to enable it to enter into a relationship with the entity-type 'choice criterion'.

The meta-model will be progressively described in the following. It will be exemplified by the F3 way-of-working and its application to the Air Traffic Control (ATC) case study performed in the project.

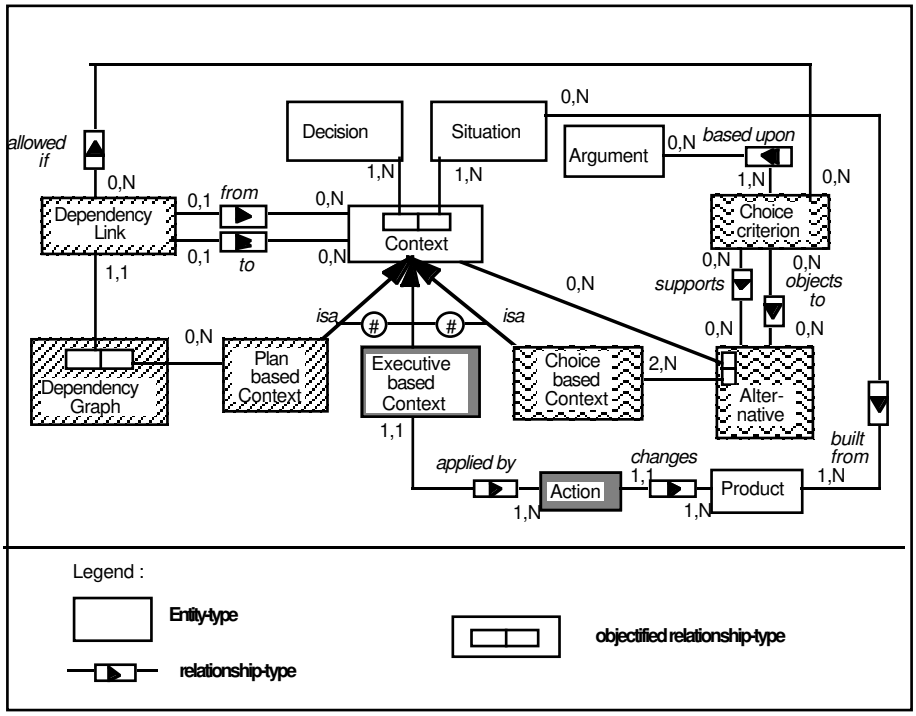

Figure 3 : Overview of the process meta-model

The central concept of this process meta-model is the one of *context* which associates a situation with a decision made on it.

A *situation* is a part of the product it makes sense to make a decision on. Situations are, most often, built from parts of the product undergoing the development process. At the beginning of the process, the product is made up of the collection of users' requirements. At the end, the output is a specification which fulfils some quality criteria. Moreover, situations can also be built from existing, reusable parts of previously developed products. [40] presents a generic model of development products and details the building of situations from its parts. Situations can be of various granularity levels; they can be either atomic like an attribute of an object class or they can be coarse-grained like the whole product under development.

A *decision* reflects a choice that an application engineer can make at a given moment in the development process. A *decision* refers to an *intention.* An *intention* expresses what the application engineer wants to achieve, it is a goal. It can be a very high level goal such as "Check_correctness_ of_current_specification" or a fine-grained one like " Define_key_of_entity".

A *context* is the association of a *situation* and a *decision* which can be made on this very situation. This association is meaningful. A decision is not sufficient in itself, it needs to be associated with the situation in which it applies. For example, it does not make sense to specialise an entity if this entity does not exist. A *situation* can be associated with several *decisions.* For instance, the situation built upon an activity in the F3 enterprise sub-model may be associated with the two decisions "Identify" and "Decompose". Similarly, a *decision* can also be made on several *situations.* The decision "Refine" can be made on a situation made up of a goal as well as an activity. The concept of context promotes the view of a decision making process where application engineers make decisions in contexts. Acting in a context corresponds to a step in the development process : in a given situation, and in order to progress in the process, the designer has to make a decision.

2.2.2 The different types of contexts

A situation exists at different levels of granularity. Further, decisions have consequences which differ from one granularity level to another. A complete understanding of the notion of a context can thus be gained by articulating the consequences of making the decision of a context on its situation. The different contexts are classified according to their consequences in the meta-model into *executive based contexts*, *plan-based contexts*, and *choice-based contexts*. We describe these three types of contexts in turn.

a) Executive-based context

At the most detailed level, the execution of the RE process can be seen as a set of transformations performed on the product under development, each transformation resulting from the execution of a deterministic action- e.g. create an activity, a goal, an actor, etc.... Such an action is a consequence of a decision made in a certain context. This leads to the introduction of the concept of an *executive-based context.*

An *executive-based context* implements a decision, its intention is realised by an action. Therefore, in the meta-model (figure 3), an executive-based context is associated with an action. An *action* performs a transformation of the product, it is the materialisation, the implementation of a decision. Performing an action changes the product under development and may generate a new situation which is itself, subject to new decisions.

Executive-based contexts establish situation-based links among contexts that we will refer to, as *correlation links. This* is modelled in figure 3 by the loop among contexts through action and situation. The term *correlation link* refers to the composition of the three following relationships in figure 3 : *applied by*, *changes* and *built on*.. Figure 4 shows two correlated executive-based contexts of the F3 way-of-working applied to the Air Traffic Control (ATC) case study. The decision to Identify the goal "Improve_air_traffic_flow" when made from scratch, leads to a new situation, namely a new goal-type component in the Enterprise sub-model on which the decision to Retype the Enterprise goal into an Information System goal can be made. Notice that <(), Identify-goal> is an executive context of the F3 way-of-working whereas <(), Identify-goal "Improve_air_traffic_flow"> is its instantiation for the ATC case study.

< ( ), Identify_goal "Improve_air_traffic_flow" >

< (Goal = "Improve_air_traffic_flow'), Retype_goal_to_ISgoal >

Figure 4: Examples of executive-based contexts

b) Choice-based context

When building a product, an application engineer may have several alternative ways to fulfil a decision. Therefore, he/she has to select the most appropriate one among the set of possible choices. In order to model such a piece of process knowledge, we introduce a second specialisation of the concept of context, namely the *choice-based context.*

A *choice-based context* corresponds to a situation which requires the exploration of alternatives in decision making. Each alternative is an approach or a strategy for the resolution of the issue being faced by the application engineer in the current situation. For example, in the F3 way-of-working, there are several ways to achieve the refinement of a goal, namely : by decomposition, linking it to an activity or through its relationship with an actor (figure 5). By definition a choice-based context offers a choice among a set of strategies, all of them achieving the same purpose. In this sense, one can look upon the choice-based context as being goal oriented.

There are two major differences between the *choice-based context* and the *executive-based context* : the first one lies in the absence of any alternatives in the latter and the second is that a choice-based context has no direct consequence on the product under development.

In the process meta-model, the various alternatives of a choice-based context are represented in the *alternative relationship* (figure 3). They are associated to choice criteria based on arguments.

A choice criterion is a combination of arguments which supports or objects to an alternative of a choice-based context. It may provide priority rules to select one *alternative* among several depending on the *arguments.* For example, in figure 4, an argument to support the alternative "Relate_goal_to_actor" is a heuristic stating that "any actor, within an organisation, must be motivated by goals". For this same choice-based context, choice criteria might suggest to select alternative (a) if the goal is expressed at a high level, in rather fuzzy terms, whereas alternative (b) could be selected if an organisational strategy to achieve the goal can be defined in terms of business activities.

According to alternative (a) the goal "Improve_air_traffic_flow" has been decomposed in the ATC case study into three conflicting sub-goals : "Reduce_delays", "Reduce_controller_stress" and "Maintain_safety_levels". The sub-goal "Reduce_delays" fits the objective of the company to increase the airport traffic; but it also increases the controllers' stress and makes more difficult the maintenance of the required safety level.

In the ATC example, alternative (b) has been applied to the goal "Maintain_safety_levels". One of the investigated organisational strategy was based on 'alert mechanisms' which could automatically display, on the controller's screen, an analytical summary of the traffic, potential risks with their type and possible decisions to make.

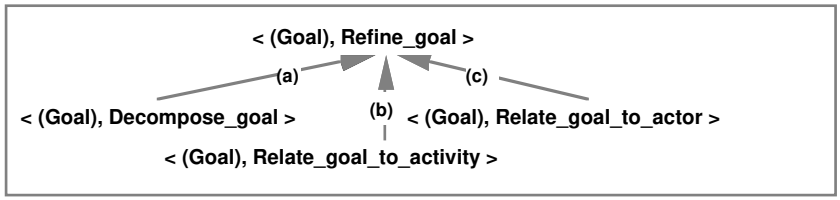

Figure 5: Example of a choice-based context

Since alternatives of a choice-based context are also contexts, contexts may share an alternative relationship (figure 3), leading to *alternative-based hierarchies of contexts*. For instance the "<(Goal), Refine_Goal>" context in figure 5, is one of the alternatives of the higher abstraction level context "<(Current_EM), Progress_in_goal_submodel>" presented in figure 6.

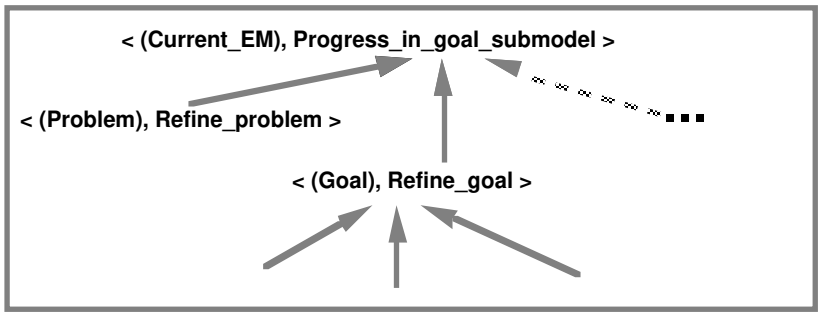

Figure 6: An alternative-based hierarchy of contexts

The alternative-based relationship among contexts allows the refinement of large-grained decisions into more fine-grained ones. It establishes refinement-based relationships that we will refer to as *refinement links*. This is a means by which the process meta-model handles the granularity problem. Notice that the term *refinement link* is just another name for the *alternative* relationship of figure 2.

The introduction in the process meta-model of alternatives and choice criteria will allow the way-of-working to support the application engineer in exploring possible strategies to resolve an issue and in selecting the most appropriate one. This alternative-based guidance leaves freedom to the engineer who can, even, make a choice which is not one of the predefined alternatives proposed by the way-of-working. This non determinism introduced in the meta-model will be useful in the capitalisation phase, to improve the way-of-working by adding new alternatives in some of its choice-based contexts.

c) Plan-based context

In order to fulfil an intention associated to a certain situation, an application engineer may be required to make a set of decisions on corresponding situations; he/she has to follow a plan. To this end, we introduce a third specialisation of context, namely, *plan-based context*. A plan-based context allows the decomposition of a context into a set of contexts.

A *plan-based context* is an abstraction mechanism by which a context viewed as a complex issue can be decomposed in a number of sub-issues. Each sub-issue corresponds to a sub-decision working on a sub-situation. The decomposition of context is another means provided by the meta-model to solve the granularity problem.

In the process meta-model the decomposition of a plan-based context into its more elementary contexts is represented (figure 3) by the relationship *dependency-graph* between context and plan-based context. The component contexts can be of any type i.e. executive, choice or plan-based contexts. For example, in the F3 methodology, for the intention named "Goal_elicitation" to be fulfilled, the two decisions "Identify_goal" and "Refine_goal" need to be made a number of times. This is modelled (figure 7) by a plan-based context, namely : "< (), Goal_elicitation >" decomposed into two contexts, namely : "<( ), Identify_goal >" executive-based context and "< (Goal), Refine_goal>", the choice-based context introduced in figure 5.

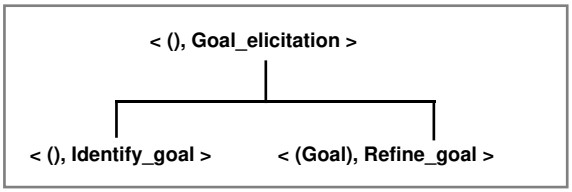

Figure 7 : Example of a plan-based context

The ordering of the contexts, within a plan, is defined by a graph named *dependency graph* (figure 3). There is one graph per plan-based context. The nodes of this graph are contexts while the links -called *dependency link*s- define either the possible *ordered transitions* between contexts or their possible *parallel enactment*. Based on arguments, a choice criterion may be assigned to a link. The choice criterion defines when the transition can be performed.

Flexibility is introduced by allowing several sets of possible parallel or ordered transitions to be defined in the same graph. For example, the graph (figure 8) associated with the plan-based context, "< (), Goal_elicitation >", describes two ways for carrying out the "Elicitation" intention depending on the preferred mode, whether depth-first or width-first. Following a depth-first mode, the sequence "Identify_goal", "Refine_goal" will be repeated, while following the width-first mode, the "Identify_goal" intention will be completed before the "Refine_goal" intention is considered. Flexibility also results from the implicit *abort* feature meaning that the plan can be aborted by the application engineer, during the process, at any moment This is another aspect of the non determinism allowed by the meta-model.

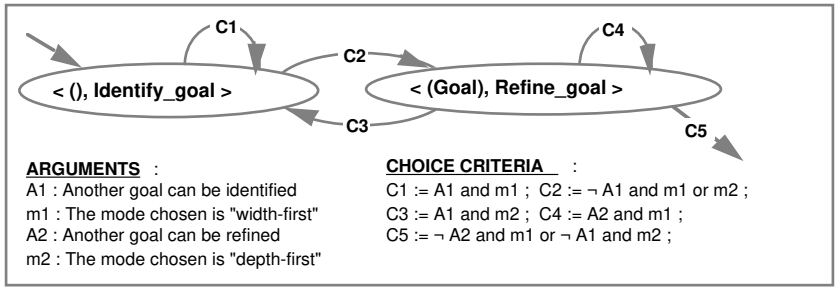

Figure 8 : Example of dependency graph associated with a plan-based context

Decomposition of contexts can be made iteratively leading to hierarchies of contexts. Figure 9 illustrates such a hierarchy as part of the F3 way-of-working to perform "Domain_understanding". We will refer to this hierarchical link as a *decomposition link*; Notice that this link corresponds in figure 3 to the composition of the *dependency graph* relationship with the *from* and *to* relationships.

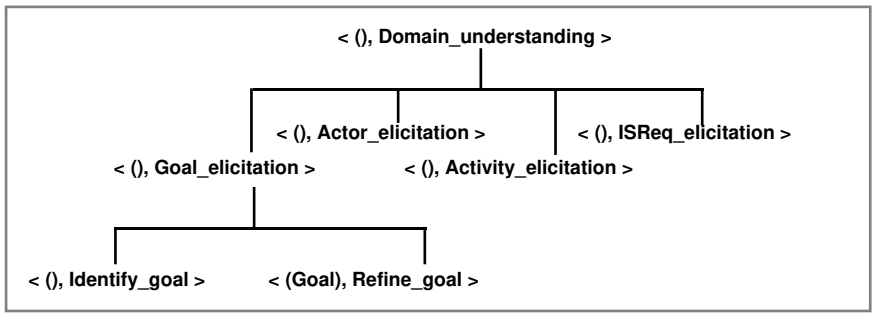

Figure 9 : Hierarchy of plan-based contexts

Plan-based contexts provide a different type of guidance than executive and choice-based contexts do. They support the application engineer in performing long term transactions, providing advice on the ordering of component activities, whereas *choice-based contexts* help in making the appropriate choice in the situation in hand and *executive-based contexts* tell how to implement the decision taken.

Each type of context influences the on-going process in a different manner : an executive context affects the product under development and generates a new situation, which itself becomes the subject of decisions; a choice-based context does not change the product but helps to further the decision making process through the refinement of an intention; a plan-based context provides the means to manage the complexity of an intention by providing a decomposition mechanism. Performing decomposition and refinement iteratively allows the application engineer to reach executable intentions and thus, to act on the product under development.

3. THE APPROACH FOR DEFINING WAYS-OF-WORKING

Based on the process meta-model presented above we present here an approach for a systematic definition of ways-of-working. The approach will be illustrated with the F3 way-of-working.

3.1 The top level view

As mentioned in the introduction, the approach merely consists of instantiating the concepts of the process meta-model i.e. defining the executive, choice and plan-based contexts and their composition, refinement and correlation linkages. As an output of this instantiation process the way-of-working built upon the notion of *context* can be globally viewed as a *network of contexts*.

The approach is itself, modelled in terms of our process meta-model. This means that the way of defining a way-of-working that we present in this section, is itself formalised in terms of any way-of-working. Indeed the process of defining a way-of-working belongs to the class of development processes

that we consider in this paper and, therefore, may be modelled in the same terms we propose to model any development process.

At the highest level the definitional approach is a plan (figure 10) composed of three contexts to make three decisions, namely : "Define_basic_block", "Build_module" and "Connect_basic_blocks". The plan is detailed by the dependency-graph presented in figure 11. We introduce the three components of the plan in turn.

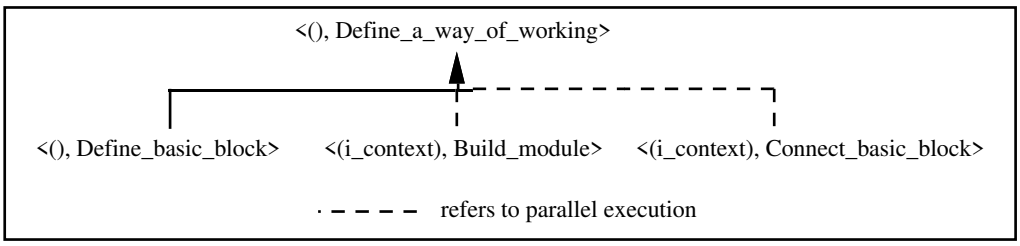

Figure 10 : A structured view of our approach

It should be clear now that due to the meta-model concepts, the basic building block of a way-of-working is an instance of context that we call *i-context.* When the "Define_basic_block" intention (figure 10) is fully achieved, the way-of-working is defined as a *collection of properly defined i-contexts.*

Contexts in the meta-model have hierarchical relationships of two different types, composition and refinement. In the way-of-working, we suggest a grouping of i-contexts based upon these links. The modules resulting from this grouping are hierarchies of i-contexts called *i-trees.* The executable purpose of the "<(i-context), Build_module>" context (figure 10) is therefore, to define the way-of-working as a *forest of i-trees*.

A non-executive i-context can be decomposed/refined into other i-contexts. Usually, one of these i-contexts is the subject of interest. However, it may happen that the application engineer wishes to deviate from these. To handle such a case, an "open-choice" is assumed to be associated with every non-executive i-context. This constitutes an open alternative in a choice-based context and an abort option in a plan based i-context.

Finally, in the meta-model, contexts have situation based relationships which are reflected in the way-of-working by introducing links between i-contexts of the same i-tree or of different i-trees. This leads to the final vision of a way-of-working as a network of i-contexts. The decision of transforming the forest into a network is called "Connect_basic_blocks"(figure 10).

The dependency-graph (figure 11) shows that there are several ways to build a way-of-working depending on the mode (width-first or depth-first) used. Irrespective of the mode, the "Define_basic_block" intention has to be achieved first but the two other intentions can be fulfilled in parallel. Following the depth-first mode, i-contexts will be defined first and then, grouped into i-trees and correlated in the network (these two steps can be performed in any order). Following the width-first mode, each i-tree definition will be followed (when possible) by its addition to an i-tree and its correlation to other i-contexts.

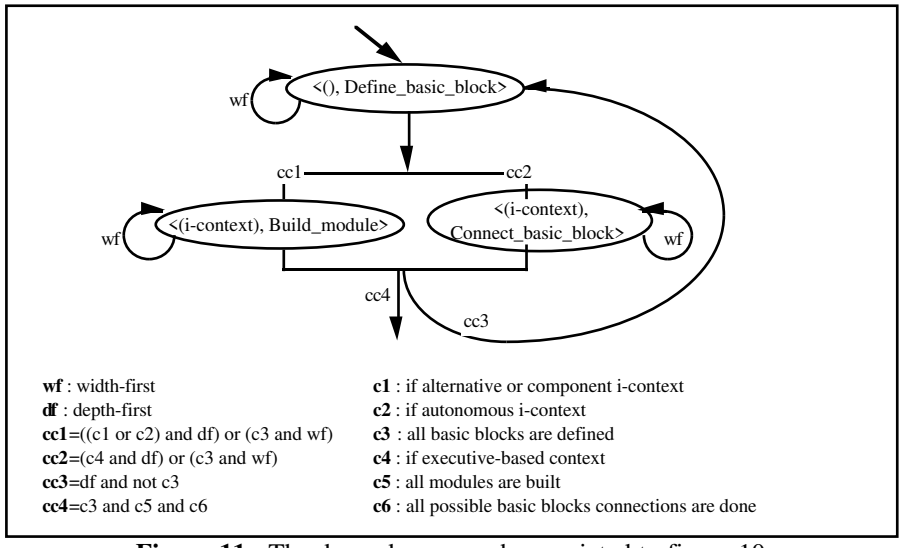

Figure 11 : The dependency graph associated to figure 10

3.2 The detailed description

In this section, we explore the approach by describing in more detail each of the three component contexts of the top level plan we have just introduced. We again use the F3 way-of-working as an example.

a) The "Define_basic_block" decision

We first present the concept of i-tree and then, the way to construct the collection of i-trees composing a way-of-working.
As an instantiation of a context, an i-context is defined by a couple (situation, decision); In the example presented in figure 12, the situation is a goal (component of the F3 Enterprise sub-model) and the decision is "Decompose_goal".

An i-context has a type, namely executive, choice or plan-based and its description includes its related elements. The description of a choice-based i-context will include for example, the list of its alternative i-contexts, their related arguments and choice criteria whereas an executive-based i-context will contain the description of the action and its impact on the product. The description of an i-context shall be looked upon as an object class specification encapsulating the description of its properties. For instance, the i-context of figure 12 is a plan-based context composed of two executive based i-contexts for respectively, creating a sub-goal and, linking it to a goal. The condition part expresses that the creation of the sub-goal and its association to the goal have to be performed in sequence while the various sequences of the former type can be performed in parallel (parallelism is formally denoted with the shuffle $\shuffle$ symbol).

i-context definition
situation : Goal
decision : Decompose_goal
type : plan-based i-context
components contexts :
- < (Goal), Create_subgoal>
- < (Sub-goal ⊔ Goal), Link_to_goal>

condition :
$\shuffle$ (<Create_subgoal$_1$> •<Link_to_goal>, <Create_subgoal$_2$> •<Link_to_goal>,...)

• : sequence $\shuffle$: shuffle

Figure 12 : Example of i-context

The way of building the required collection of i-trees for a given way-of-working is presented in figure 13 as the decomposition of the plan-based context "< (), "Define_basic_block>" into two contexts, namely "< (), Identify_i-context>" and "<(i-context), Define_i-context>".
The latter is a plan-based context exclusively composed of executive ones (describing the actions to be performed for achieving the complete description of the i-context); the former is a choice-based context suggesting two different srategies for identifying i-contexts : the one driven by situations (each atomic or complex concept participating to the product construction is, for instance, a situation subject to primitive decisions) and the other one, driven by intentions (one can relate here to a generic collection of modelling and requirements elicitation decisions).

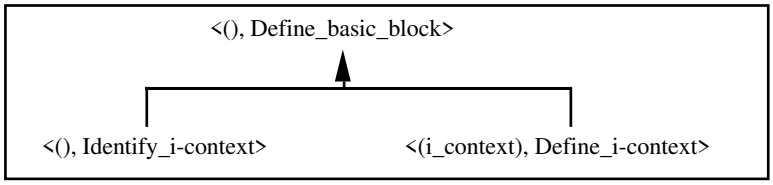

Figure 13 : The plan-based "Define_basic_block" context

b) "The Build_module" decision

In the meta-model, contexts may participate in hierarchical relationships through either composition or refinement links. A composition link relates a plan-based context to its components whereas a refinement link associates a choice-based context to one of its alternatives. Choice-based contexts and plan-based contexts are built over contexts whereas executive-based contexts are not. According to the foundational lexicon presented by Humphrey and Feiler [19] executive-based contexts are 'process steps' (they have no visible substructure) while choice-based contexts and plan-based contexts are 'process elements' : they are built from contexts like 'process elements' are built from steps.

This leads us to consider hierarchies of contexts, resulting from the iterative application of either decomposition or refinement link, as meaningful *modules* of the way-of-working. We call these hierarchies i-trees. An *i-tree* has a root which is, either a choice-based context, or a plan-based context. It has a variable number 'n' of levels (n>=2). The more global the root intention is, the more levels are required to support its decomposition and/or refinement. One may notice that the nodes of an i-tree are contexts of any of the three types : executive, choice or plan. Executive-based contexts are leaves of i-trees, and if the way-of-working is completed, all the leaves should be contexts of the executive type. The intermediate nodes can be either choice or plan based contexts.

A way-of-working is normally composed of a collection of i-trees, it is a *forest of i-trees*. The existence of multiple i-trees in a unique way-of-working comes from the independence of some contexts due to the independence of either situations or decisions. For example, a number of executive decisions which can be taken from scratch, participate to contexts which are roots of i-trees. This can also occur to choice-based contexts and plan-based contexts. However, at the same time, such contexts can be embedded in a more global i-tree as sub-i-trees. As an illustration, figure 14 and figure 15 present respectively, part of the main i-tree of the F3 way-of-working and a partial list of its i-trees, some of them being also included in the main i-tree. For the sake of readability, i-trees include only decision names.

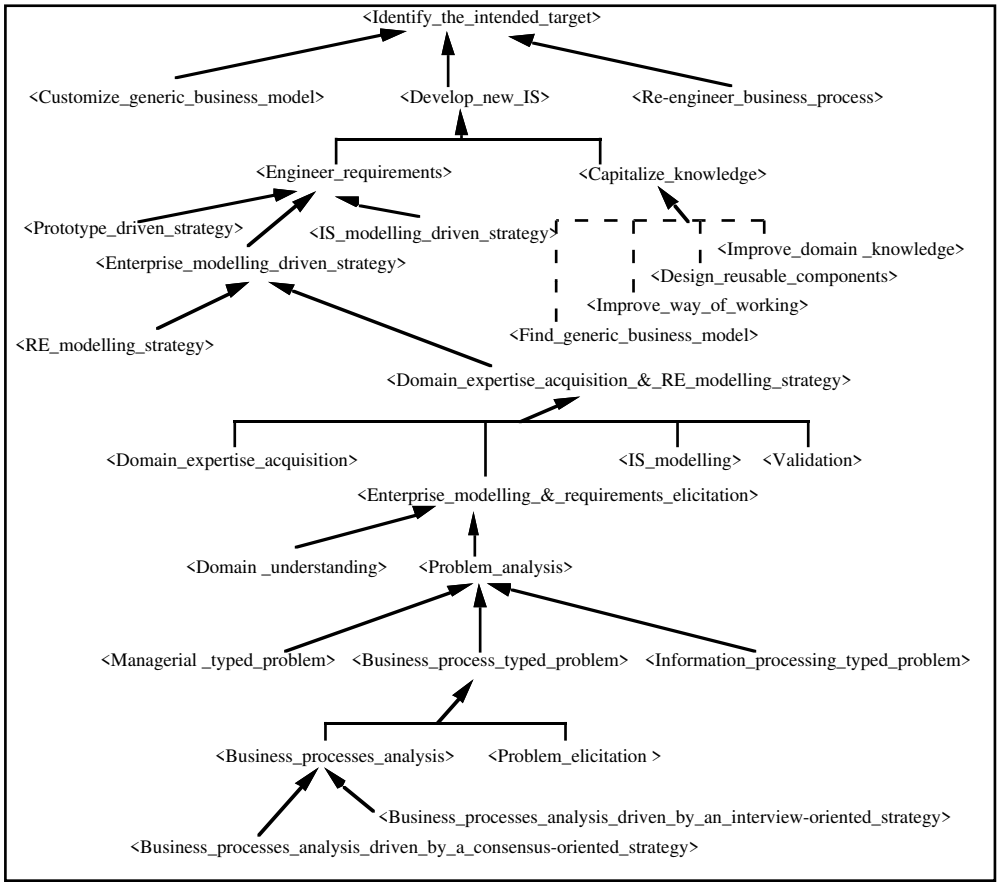

Figure 14 : The main F3 i-tree

We comment on the 10 levels of the i-tree with the purpose of making our view of ways-of-working more concrete. The example is intended to show how the combination of plan and choice-based contexts can encapsulate methodology knowledge in a rather powerful but easy to read manner. It tries also to demonstrate the role of arguments and choice criteria in capturing the experimental and heuristic knowledge of professionals. For reasons of space, we have restricted ourselves to one example of arguments (figure 16).

- Identify_the_intended_target
- Engineer_requirements
- Capitalize_knowledge
- Domain_expertise_acquisition_&_RE_modelling_strategy
- RE_modelling_strategy
- Enterprise_modelling_&_requirement_elicitation
- Domain_understanding
- Problem_analysis
- IS_modelling
- Domain_expertise_acquisition
- Validation
- Refine_goal
- Decompose_goal
- Create_actor
- Retype_goal_to_IS_goal

.....

Figure 15 : Some roots of F3 i-trees

The top level decision in figure 14 is made only once per project. It aims at selecting the guidance appropriated to the domain area : generic business model construction, IS development or business process re-engineering. Level 2 for IS development suggests a plan composed of

"Engineer_requirements" and "Capitalize_knowledge". F3 is indeed, tailored to capitalise from experiences in different ways (Design reusable components, Improve way-of-working, etc..) which can be performed in parallel (dotted lines of the plan-based context "Capitalize_knowledge"). The way-of-working offers the choice of three different strategies for "Engineer_requirements" which are argumented and compared in the body of the corresponding choice-based context. The "Enterprise_modelling_ driven_strategy" can itself be followed in two different ways with or without the development of a domain knowledge base. In the former case, the way-of-working proposes a plan-based decision making process composed of "Domain expertise_acquisition", "Enterprise_modelling_requirements_elicitation", "IS_modelling" and "Validation". Enterprise modelling can in turn, serve two different purposes, namely "Domain_understanding" and "Problem_analysis" for which it exists three alternatives with different guidance. The one introduced at level 8 of the i-tree is a plan-based guidance for "Business _process_analysis" and "Problem_elicitation". Finally level 9 offers a choice of strategy for driving the group of people who will perform the problem analysis; it can be consensus driven (see figure 15 for arguments) or interview driven.

The Business_process_analysis_driven_by_consensus-oriented_strategy aims at modelling the requirements with the active participation of the clients and the engineers leading to collective, consensus-oriented decisions.
choice-criterion((A1 and A2) or A3) and (A4 and not A5)

list of arguments :
A1 : the intention is more to elicit requirements than to capture requirements
A2 : consensus among actors of the Enterprise is required
A3 : lack of communication & social problems within the enterprise may imply a risk for the success of the project
A4 : enough persons are involved with good group management skills
A5 : the number of actors to involve is too high

Figure 16 : The choice criterion associated to "consensus-oriented strategy"

The way of building the forest of i-trees is presented in figure 17 as the decomposition of the plan "< (i-context), Build_module>" into two contexts, namely "<(i-context), Connect>" and "<(i-tree), Create>". The first one is a plan-based context to repeatedly connect either a plan-based context or a choice-based context to another context if they are either alternative or component of the latter. I-trees are rather easily constructed this way. The second context aims at supporting the creation of autonomous i-trees. The decision to create an i-tree is based on the evaluation of the independence of the root decision.

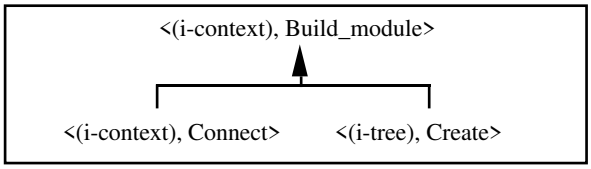

Figure 17 : The plan based "Build_module" context

c) The "Connect_basic_blocks" decision

In the meta-model, executive-based contexts generate links among contexts through the generation of new situations as consequences of actions which are subjects to decision making. We called these links correlation links. Instantiation of correlation links during the construction of the way-of-working leads to establish connections among i-contexts belonging either to the same i-tree or to different i-trees. An example of connection in the F3 methodology between the two contexts "<(), Identify_goal>" and "<(Goal), Retype_goal_to_ISgoal>" can be derived from figure 3 : the consequence of the decision to "Identify_Goal" is a new goal in the Enterprise sub-model (the goal to "Improve_air_traffic_flow" for example) which is a situation matching the situation part of the context "<(Goal), Retype_goal_to_ISgoal>".

With the introduction of correlation links, the way-of-working is now a *network of i-contexts.* The construction of the network results of the repetitive decision to "Connect_basic_blocks" for each executive context which generates a situation which is itself, part of one or more other contexts.

4. ILLUSTRATION OF PROCESS GUIDANCE

In this section we sketch how guidance is provided by the way-of-working. The sketch is based on the ATC case study.

4.1 The keys of process guidance

It should be clear now, that the way-of-working is suggesting a contextual approach in which requirements engineering consists of making decisions in context. Guidance within this approach is provided :

- to support the application engineer in making decisions in the situations he is faced to; this is a *local guidance* provided at certain points in time.
- to help in monitoring the flow of decisions allowing an efficient progression in the engineering process; this is a *global guidance* occurring over time.

Local guidance is provided by an i-context. Global guidance is based upon links between i-contexts. Thus, the guided requirements engineering process can be viewed as a sequence of steps, each of them relating to one context. Local guidance occurs in one step while global guidance is visible in a sequence of steps.

The guidance provided by an i-context depends on its type. An executive i-context tells which action to perform now; a plan-based i-context, if exclusively composed of executive i-contexts, plays a similar role; it describes the set of ordered and/or parallel actions to be performed to fulfil the intention of the plan. A choice-based context supports the engineer in the deliberation process leading to either the choice of one alternative among a set of possible choices or to the open choice given in every i-context. A plan-based context helps in selecting either the context in which to work or the abort option.

The progression from one context to another one is supported by the knowledge encapsulated in i-context links. Hierarchical and correlation links play here a different role. Correlation links are useful after the execution of one (in case of an executive context) or several actions (if an executive plan), to decide on which context to work in the next step. Correlation links offer choices to the engineer who can follow one of the suggestions or not. Hierarchical links help either in the refinement or in the decomposition of an intention expressed at a level which requires to be made more detailed in order to be implemented. Refinement links are used in a top-down fashion, from the root of the i-context to its executive leaves. Decomposition links are first followed from the root of the i-tree in a top-down manner till an executive context is reached and, then, in a bottom-up fashion to progressively ensure the complete execution of the component plans encountered as nodes of the i-tree.

Even in circumstances where the application engineer does not wish to follow the prescriptions of the way-of-working at the moment they are proposed (some of these have been mentioned above), guidance can still be provided based on a different exploitation of the knowledge encapsulated in the way-of-working. For example, applying a pattern matching algorithm, it is always possible to select all situations in the current state of the product under development on which guidance can be provided. Similarly, all situations matching a certain intention or its derived intentions can be displayed. Thus the guidance provided is a combination of two guidance modes of the way-of-working :

- the prescriptive mode in which i-trees and i-contexts are systematically followed, and
- the engineer-driven mode in which the application engineer deviates from the prescriptive mode. The process knowledge acquired by the engineer is capitalised, for example, to take advantage of the "open-choice" referred to earlier.

4.2 The ATC scenario

Figure 18 sketches the prescriptive guiding mode which has been applied to the ATC case study.

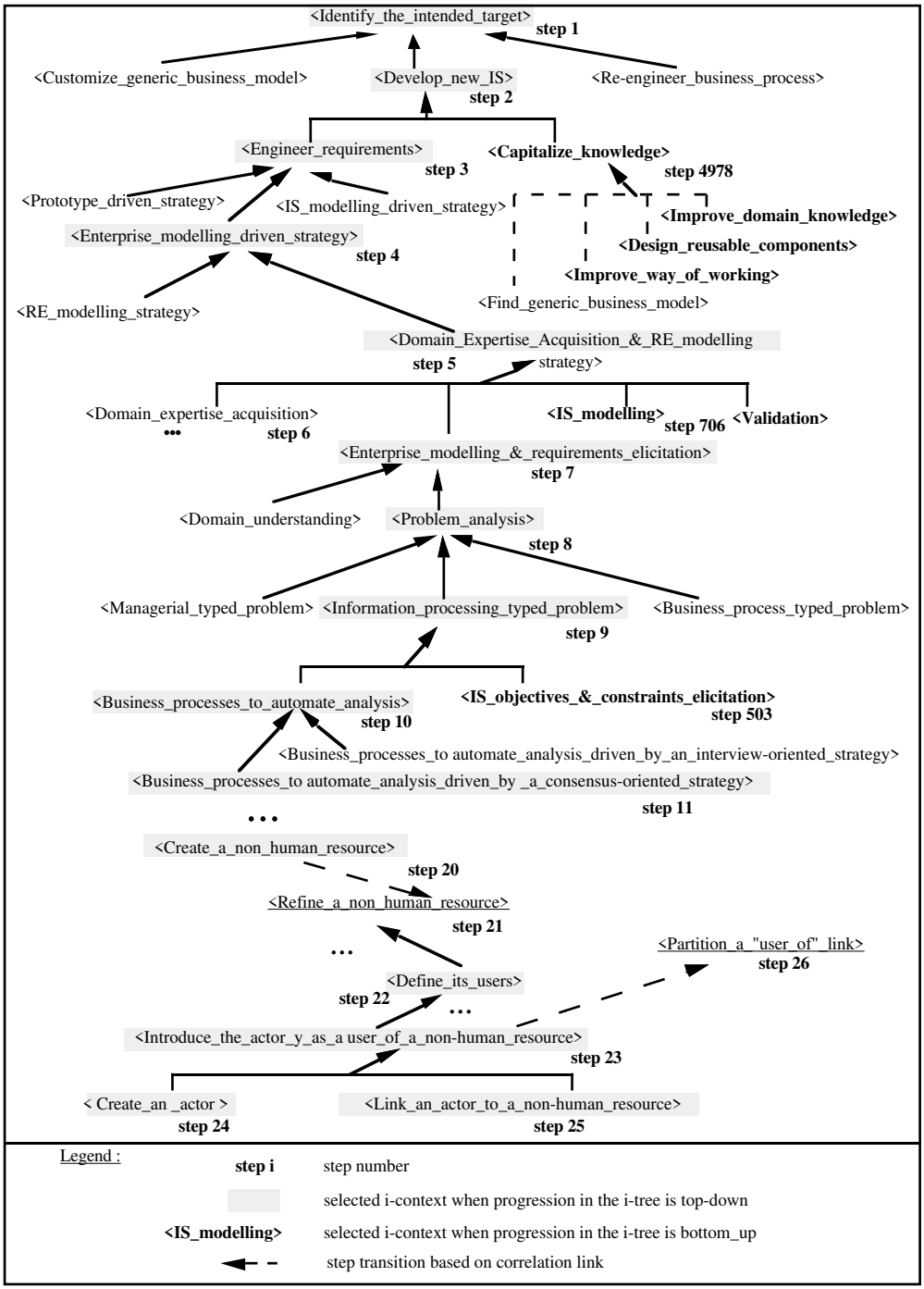

Figure 18 : Part of guided process of the ATC case study

The part of the guided process illustrated in figure 18 results mainly, from the use of the F3 i-tree presented in the previous section and, partly from some complementary i-trees.

The steps of the process are numbered in ascending order starting from the selection of the target <Develop_new_IS>. Indeed, the objective of the project was to develop a new ATC sequencer able to automate the activities of planning, scheduling and sequencing aircraft landings in airports.
The first 20 steps illustrate the guidance provided in order to progress from a high level intention, "investigate the problem of an automated sequencer", to one concrete action (define the sequencer as a non human resource) contributing to the achievement of this intention. The selected path in the i-tree led to the making of choices (for instance to develop expert knowledge on this domain : step 5) and to follow plans (steps 6 and 7 for instance).
The progression from step 20 to step 21 is of another nature, it is suggested by a correlation link. A similar situation occurs in the step transition from 23 to 26.

Leaves of i-trees are executive contexts. This is first illustrated in steps 24 and 25, both of them corresponding to the execution of actions, here consisting of the creation of the controller as an actor linked to the sequencer which has been identified (in step 20) as a non human resource.

The bottom-up mode for following a plan is illustrated in step 503 (this number corresponds to the number of real steps which were necessary to perform entirely the "Business_processes_ to_automate_analysis" intention preceding the "IS_objectives_&_constraints_elicitation" intention of the same plan "Information_processing_typed_problem". This shows that the complete achievement of the decision called "Business_processes_to_automate_analysis" required the treatment of 402 contexts. Following a plan implies backtracking in the i-tree from a leave to a highest level in the hierarchy. The number of traversed levels increases as the process proceeds. For instance in the ATC case study, the last indicated backtrack was made to "Capitalize_knowledge" in step 4978.

Choices in the top levels of the hierarchy have important consequences. For example the decision to develop the expert knowledge base for the ATC has been a 6 man/months investment. This highlights the essential role played by choice-based contexts as well as the need for well defined arguments. In the former case, arguments take into account the expertise of the expert system developers, the results of empirical studies evaluating the role of domain knowledge, quantitative measures of risks resulting of the non-development of the expert base and qualitative factors as well.

5. CONCLUSION

We first summarise the key aspects of our approach for defining ways-of-working, then, show how the six requirements for guidance centred process models are fulfilled and finally draw some directions for future work.

Summing up the approach

The process meta-model presented in the paper is the theoretical foundation of our ways-of-working. It identifies the notion of context as the basic building block of development processes. A context is composed of a situation and a decision. It models the very basic fact of engineering which is to make a contextual decision i.e. a decision which depends on and is adequate for, the situation in hand. Moreover, in order to cope with the large variety of situations and decisions encountered in the development process, the meta-model proposes a sub typing of contexts into executive, choice and plan-based contexts. A choice-based context corresponds to a situation which requires the exploration of alternatives in decision making for its treatment, whereas an executive-based context is only concerned with the implementation of the decision through actions. Finally, a plan-based context organises decision making into a plan. One of the key properties of the meta-model is to permit combining the three kinds of contexts. Thus, alternatives of a choice-based context can be choice, plan or executive contexts. We believe this is a powerful feature for modelling complex process situations which is not supported by most existing process models. Since alternatives of choice-based contexts and components of plans are contexts too, the meta-model provides an elegant solution to solve the granularity problem and to describe hierarchies of contexts. Finally through the notions of argument and choice criteria attached to alternatives, and the one of dependency graph associated to plan-based contexts, the meta-model is able to capture heuristics and experimental process knowledge.

We claim that any way-of-working related to any methodology dedicated to system development can be described as a network of context-trees. A context-tree called i-tree gathers all associated i-contexts

(instances of contexts) together by either a composition link (through plan-based context) or refinement link (through choice-based context), the leaves of the i-tree being executive based i-contexts. An i-tree encapsulates an interesting piece of process knowledge. It is a process chunk which provides support to solve the issue raised by the root i-context. This support comes in two ways:
- by hierarchical links which help to deal with the decomposition or the refinement of a complex decision.
- by the accumulation of a large amount of heuristical knowledge in dependency graphs, arguments and choice criteria. To be able to express heuristics is an important improvement in way-of-working definitions since it allows the capitalisation and sharing of experiences.

The way-of-working is viewed as a network of context-trees because it integrates correlation links among i-contexts. A correlation link (through executive context) provides knowledge to establish situation-based connections in the process. It is a useful piece of knowledge for guiding the process flow (suggesting what can possibly be done next) which is not provided by existing ways-of-working. Thanks to the meta-model :
(a) ways-of-working have a good level of genericity,
(b) they are defined in a modular manner with i-contexts as basic building blocks and i-trees as modules and are consequently easy to evolve and change, and
(c) they guide requirements engineers locally (within one context) and globally as well (to monitor the context flow) in a flexible manner.

How are the requirements fulfilled

Figure 19 summarises how the requirements for guidance centred process models are fulfilled by ways-of-working constructed following our approach. The definitional approach based on instantiations of the generic process meta-model is essential for guaranteeing the genericity of ways-of-working and their independence from specific methodologies. The generic concept of a context with three specialisations contributes to the genericity of ways-of-working and satisfies the requirement for variable granularity. Process model grains range from broad life-cycle descriptions, such as the ones suggested by the waterfall model, to very detailed ones, such as the rule "compile program after modification". The i-context and i-trees, which are used in the expression of ways-of-working, have a very modular form. This eases their evolution and improvement. Finally, the decision and situation based paradigm seems to allow the capture of heuristic and experimental knowledge necessary to guide human intellectual activities taking place during system development.

requirements of process models	**Achieved through**
granularity	decomposition and refinement of contexts
paradigm	situation and decision oriented
genericity	the meta-model approach
guidance	choice-criteria in choice-based context and in dependency graphs non-determinism feature in context
self-improvement	capitalising process experiences
evolution support	i-tree and network of i-trees

Figure 19 : Achievement of the criteria presented in the introduction.

Future work

In the paper, we have taken examples from the F3 methodology and its application to the ATC case study developed in the project. Other methodologies have been described with our approach (e.g. OOA, OMT, ER, SASD among others). We are currently working on the implementation of the framework sketched in section 1. The first result clearly shows the effectiveness of our approach : the application engineer is truly guided. However, the problem of a distributed process guidance has to be tackled. Another important direction for future research is to address the representation of the managerial aspects of the process and to combine them with the engineering ones.

REFERENCES

[1] 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.

[2] 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.

[3] C. Batini, S. Ceri, S. Navathe, *Conceptual Data Base Design : an ER approach*, Benjamin/Cummings (Pub.), 1992.

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

[5] T.M.A. Bemelmans, *Bestuurlijke informatiesystemen en automatisering*, Stenfert Kroese, Leiden, The Netherlands, 1987.

[6] 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.

[7] J. Blank, M.J. Krijger, *Evaluation of methods and techniques for the analysis*, design and implementation of information systems, Academic Service, The Hague, The Netherlands, 1982.

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

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

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

[11] J.R. Bubenko, J.A. Bubenko, *Information System Methodologies - A Research View*, in Olle T.W., Sol H.G., Verrijn-Stuart A.A. (Eds.) : "Information Systems Design Methodologies : Improving the practice, North-Holland, Amsterdam, The Netherlands, 1986, pp 289-318.

[12] P.P.S. Chen, *The Entity-Relationship Model : Towards an unified view of data*, ACM Transactions on Database Systems, Vol 1, N°1, March 1976.

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

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

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

[16] M. Dowson, *Software Process Themes and Issues*, IEEE int. conf. , 1993.

[17] M. Dowson, *Towards Requirements for Enactment Mechanisms*, European worshop on Software Process Technology WSPT'94, 1994, pp90-106

[18] L.J.B. Essink, *A Conceptual Framework for Information System Development Methodologies*, Memorandum INF-86-34, University Twente, Enschede, The Netherlands, 1986.

[19] P.H. Feiler, W.S. Humphrey, *Software Process Development and Enactment : Concepts and Definitions*, Proc. 2nd Int. Conf. on "Software Process", 1993.

[20] C. Fernström, L. Ohlsson, *Integration Needs in Process Enacted Environments*, Proc. 1st International Conference on the Software Process", IEEE computer Society Press, October 1991.

[21] M. Franckson, C. Peugeot, *Specification of the Object and Process Modeling Language*, ESF Report n° D122-OPML-1.0, 1991.

[22] G. Grosz, C. Rolland, *Using Artificial Intelligence Techniques to Formalize the Information Systems Design Process*, Proc. Int. Conf. on "Databases and Expert Systems Applications", 1990.

[23] A.H.M. Hofstede, T.F Verhoef, G.M. Wijers G.M., S. Brikkemper S., *The SOCRATES project,* proc. of NGCT'90, pp X1-1-X1-7, Noordwijkerhout, The netherlands, April 1990.

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

[25] 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.

[26] 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.

[27] M. Jarke, *Requirements Engineering : An Integrated View of Representation*, Process and Domain", Proc. 4th European Software Engineering Conference, Springer Verlag (Pub.), 1993.

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

[29] G.E. Kaiser, N.S. Barghouti, P.H. Feiler, R.W. Schwanke, *Database Support for Knowledge-Based Engineering Environments*, IEEE Expert, 3(2), 1998, pp18-32.

[30] J. Lonchamp, *A Collaborative Process Centered Environment Kernel*, Proc. 6th Int. Conf. on "Advanced Information Systems Engineering" (CAISE), Springer Verlag (Pub.), 1994.

[31] P. Loucopoulos, R. Zicari, *Conceptual Modeling, Database and Case*, Wiley (Pub.), 1992.

[32] M. Lubars , C. Potts, C. Richter, *A Review of the State of the Practice in Requirements Modeling*, Proc. Int. Symposium on "Requirements Engineering", 1993.
[33] K. Lyytinen, K. Smolander, V-P. Tahvainen, *Modelling CASE Environments in sytems Work*, CASE'89 conference papers, Kista, Sweden, 1989.
[34] P. Martiin , M. Rossi, V-P. Tahvainen, K. Lyytinen, *A comparative review of CASE shells - a preliminary framework and research outcomes*, published in Information and Management, 25, 1993, pp11-31.
[35] P. Martiin, *Methodology Engineering in CASE shells : Design Issue and current Practice*, PhD thesis, Computer science and information systems reports, Technical report TR-4, 1994.
[36] 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.
[37] C. Potts, *A Generic Model for Representing Design Methods*, Proc. 11th Int. Conf. on "Software Engineering", 1989.
[38] C. Rolland, C. Cauvet, *ALECSI : An Expert System for Requirements Engineering*, Proc. 3th Int. Conf. on "Advanced Information Systems Engineering" (CAISE), Springer Verlag (Pub.), 1991.
[39] C. Rolland, *Modeling the Requirements Engineering Process*, Information Modelling and Knowledge Bases, IOS Press, 1993.
[40] J.R. Schmitt, *Product Modeling in Requirements Engineering Process Modeling*, IFIP TC8 Int. Conf. on "Information Systems Development Process", North Holland (Pub.), 1993.
[41] Sol H.G.: "Kennis en ervaring rond het ontwerpen van informatiesystemen", Informatie, Vol. 27, N°3, 1985.
[42] Sol H.G.: "Information Systems Development : A Problem Solving Approach", in Proceedings of 1988 INTEC Symposium Systems Analysis and Design : A Research Strategy, Atlanta, Georgia, 1988.
[43] Y. Sugiyama, *Producing and Managing Software Objects in Process Programming Environment OPM*, IEEE Asia-Pacific Software Engineering Conference (APSEC), 1994, pp268-277.
[44] *Tempora esprit project : final report*, 1994.
[45] K. Tominaga, T. Tokuda, *Constraint-Centered Descriptions for Automated Tool Invocation*, IEEE Asia-Pacific Software Engineering Conference (APSEC), 1994, pp92-101.
[46] G.M. Wijers, *Modeling Support in Information Systems Development*, PhD Thesis, Thesis Publishers Amsterdam, 1991.
[47] Wilenski, *Planning and Understanding*, Addison Wesley (Pub.), 1983.
[48] J. D. Wynehoop, N. L. Russo, *System Development methodologies: unanswered questions and the research-practice gap*, Proc. of 14th ICIS (eds. J.I. DeGross, R.P. Bostrom, D. Robey), Orlando, USA, 1993, pp. 181-190.