A Decision Making Pattern for Guiding the Enterprise Knowledge Development Process

Colette Rolland, Selmin Nurcan, Georges Grosz

▶ **To cite this version:**

Colette Rolland, Selmin Nurcan, Georges Grosz. A Decision Making Pattern for Guiding the Enterprise Knowledge Development Process. Journal of Information and Software Technology, 2000, pp.313 - 331. hal-00707096

HAL Id: hal-00707096
https://hal.science/hal-00707096

Submitted on 16 Jun 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Decision Making pattern for Guiding the Enterprise Knowledge Development Process

Colette Rolland, Selmin Nurcan, Georges Grosz

Université Paris 1 - Panthéon - Sorbonne
Centre de Recherche en Informatique
90, rue de Tolbiac 75013 Paris, France
email: {rolland, nurcan, grosz}@univ-paris1.fr

Abstract

During enterprise knowledge development in any organisation, developers and stakeholders are faced with situations that require them to make decisions in order to reach their intentions. To help the decision making process, guidance is required. Enterprise Knowledge Development (EKD) is a method offering a guided knowledge development process. The guidance provided by the EKD method is based on a decision making pattern promoting a situation and intention oriented view of enterprise knowledge development processes. The pattern is iteratively repeated through the EKD process using different types of guiding knowledge. Consequently, the EKD process is systematically guided. The presentation of the decision making pattern is the purpose of this paper.

Keywords: Decision Making, Enterprise Knowledge Engineering, Guidance, Change Process, Re-use.

1. Introduction

Software engineering and requirement engineering methods were classically focused on the product aspect of systems development and have paid less attention to the description of formally defined processes which could be supported by CASE environments.

Existing approaches to *enterprise knowledge modelling* can be classified into two categories. In the first category, an organisation is represented as a set of inter related elements satisfying collaboratively common objectives [4], [8], [9]. For instance, VSM [8] allows us to model an organisation as a set of "viable" sub-systems representing respectively the operation, co-ordination, control, intelligence (reasoning, analysis) and politics (strategy) aspects of an organisation. In the second category, the focus is given to developing different views of the organisation dealing respectively on actors, roles, resources, business processes, objectives, rules, etc. [2], [6], [13], [14]. The development processes suggested by these approaches are descriptive, remain at a very high level of abstraction and do not provide effective guidance.

Existing approaches to *organisational change management* propose sequences of activities in order to change the organisation from its current state to a desired future state. Many of them are adaptations and/or extensions of the three steps proposed by Lewin [18]: (1) to de-freeze business processes as a sign of preparation and disposition to change, (2) to

implement the change, (3) to standardise (freeze) new processes, structures and systems. Other authors have extended this approach by describing more precisely these three steps and propose again sequential processes [10], [16]. In all these approaches, change process descriptions are coarse grained and described as sequences of steps. They have not associated product models.

Clearly, there is a high need for methods and tools which offer process guidance to provide advice on which activities are appropriate under which situations and how to perform them [7], [12], [36], [28], [29] to handle change management. In [30] we proposed a method, namely Enterprise Knowledge Development (EKD) which intends to provide such guidance. The EKD method has been applied, in the context of the ESPRIT project ELEKTRA[1] [37] for re-organising electricity companies and designing new solutions [20], [21].

The EKD framework is composed of three complementary elements :
(1) a set of models used for describing the system to be constructed and the organisation in which it will operate,
(2) a process supporting the usage of concepts,
(3) a set of tools supporting the EKD process.

The focus of this paper is given to point (2), the readers shall refer to [3, 20] for point (1) and [11] for point (2).

The EKD process supports the enterprise knowledge development in a systematic manner. It provides advice on what should be considered during the design process (goals, roles, etc.), why and how it should be analysed (goal decomposition, role dependency study, etc.) following some relevant techniques (brainstorming, SWOT analysis, etc..

We distinguish three levels in process modelling:

- At the instance level, process traces are recorded. A process is a description of the route followed to construct a product. The output of a process is a *product*, it can be a requirements specification, a conceptual schema or a set of business goals. A process and its related product are specific to an application.
- At the type level, process models are defined. A process model is an anticipation of what the process will look like. A process is then, an instantiation of the process model. A process model is specific to a method.
- At the meta-type level, we define the generic concepts used to represent any process model. Along with their relationships, those concepts constitute the process meta-model. Process models are therefore, instances of the process meta-model. A process meta-model is method independent.

The EKD process is a guided process made of steps resulting each in the application of the same *pattern for decision making*. This paper is dedicated to the presentation of the *EKD decision making pattern* which aims at supporting the EKD engineer while performing the

[1] This work is partially supported by the ESPRIT project ELEKTRA (N° 22927) funded by the EEC in the context of the Framework 4 programme.

decision making process when studying the impact of change in an organisation. It is organised as follows. Section 2 sets the aim and objectives of the pattern and presents the concepts used to represent it. Section 3 illustrates the use of the pattern through different examples, putting forward the different types of guidance it provides. Finally, section 4 gives an overview of the method specific guidelines used in the EKD framework, and illustrates some of them. Section 5 concludes this paper.

2. The EKD decision making pattern

2.1. The EKD method and the underlying EKD models

The purpose of the EKD method is to provide a clear and unambiguous picture of (1) how the enterprise functions currently, (2) what are the requirements for change and the reasons for change, (3) what alternatives could be envisaged in order to meet these requirements, and (4) what are the criteria and arguments for evaluating these alternatives [3], [21].

The models underlying the EKD method are based on the use of Enterprise Models [2] which purpose is to represent various views of an enterprise. A detailed presentation of the EKD Models can be found in [3] and [20]. The *goal model* aims to represent the goals of the enterprise. Its purpose is to describe what the enterprise wants to achieve or to avoid. The *business rules model* is used to define business rules consistent with the goals model. Business rules can be seen as operationalisation of goals. The *object model* is used to define the enterprise entities, attributes and relationships. The *actor/role* model aims to describe how actors are related to each other and also to goals. The *role/activity model* is used to define enterprise processes, the way they consume/produce resources. The *information system model* is used when the EKD method is applied to define the requirements for an information system. In this model, the focus is the computerised system which has to support the goals, the processes and the actors of the enterprise as defined in the previous models. These set of EKD models can be visualised in three levels of concern as shown in figure 1: business objectives, business processes, information systems [20]. In using the EKD method, one can start at any level and move on to other levels, depending on the situation.

The EKD method can be used for the purposes of both business engineering and information system engineering [20] allowing:

(a) business process reengineering (from business processes level to the business objectives for change [31]) ;

(b) reverse engineering (from legacy information systems to the information system level which may be than used to model the business processes level [15]) ;

(c) forward engineering (from business objectives to business processes and from business processes to information systems development).

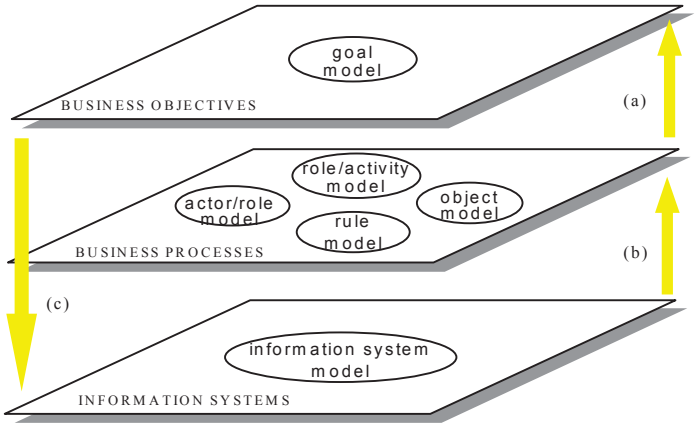

Figure 1: *EKD models*

2.2. Aim and scopes of the decision making pattern

The EKD process aims at constructing a description of the Enterprise Knowledge in terms of EKD models. The EKD decision making pattern is the key element underlying the EKD process. It aims at providing a reasoning mechanism to support the complexity of change processes. It is based on two assumptions. First, change processes are non determinist and unpredictable and therefore, must be regarded as decision making processes which cannot be planned in advance. However, changes can be sometimes performed by analogy to other changes, particularly if they occurred in the same domain (e.g. Air Traffic Control (ATC), electricity distribution, etc.). This supports the belief that it exists a corpus of both generic and domain specific method knowledge (i.e. knowledge on how to proceed for change) that can be looked after, identified and capitalised from experiences.

Based on these assumptions, the decision making pattern sets the following objectives :

- to give to the EKD engineer the freedom to move forward in a dynamic manner depending on the situation he/she is faced to and the intention he/she wants to achieve,
- to provide guidance based on the capitalised generic and specific method knowledge,
- to support the decomposition of complex processes.

In order to meet these objectives, the decision making pattern :

(1) promotes a contextual perspective in decision making where decisions are made with a clear understanding of their organisational context. The contextual approach relies on the notion of *context* which couples an *intention* to be achieved by the EKD engineer and the organisational *situation* which justifies this intention and clarifies the context in which the decision has to be made,

(2) offers guidance at three levels: the generic level, the EKD level and the domain specific level. Each corresponds to a specific type of method knowledge. *Generic knowledge* is characteristic of any decision making process performed following the contextual paradigm. *EKD knowledge* is typical to the process models defined to work with the EKD concepts and models. *Domain specific knowledge* is defined at the instance level.

Examples of such knowledge are Air Traffic Control knowledge or Command and Control systems development knowledge,

(3) supports the view of an *iterative* and *dynamic* process. The process is iterative as it results from the repetition of the decision making pattern which is applied iteratively to any situation arising in the process and requiring decisions to be made. Its dynamicity comes from the fact that the EKD engineer is free to dynamically choose the context he/she wants to work on. Therefore, every step corresponds to one application of the EKD decision making pattern and steps are dynamically linked.

2.3. Definition

The EKD decision making pattern is a *reasoning mechanism* supporting decision making using a *library of guidelines* and an *enactment mechanism* (see figure 2). The enactment mechanism plays a double role. First, it helps in the retrieval of the appropriated guideline from the library supporting decision making at any stage of the process, i.e. in the current situation at hand as defined in the next paragraph. Second, it is used to guide the decision making according to the selected guideline (see [33] for details). As shown in figure 2, the input and the output of the enactment mechanism are contexts. The decision is made in a given context (the input), may affect the product under development and generate one or more new situations motivating new decisions to be made (the output contexts).

The decision making pattern is tailored to provide guidance in all cases. In some cases, the pattern offers a domain specific guidance. This happens when the library contains domain specific knowledge which matches the current context of the EKD process. For instance, the library may contain a guideline in the ATC domain suggesting to decompose the goal "*Minimise risk of befalling aircrafts*" into the two sub-goals "*Maintain separation standards*" and "*Decrease risk of human errors*". If during the process of an air traffic control project the intention of operationalising the goal "*Minimise risk of befalling aircrafts*" is raised, the library will be able to offer a domain specific guideline.

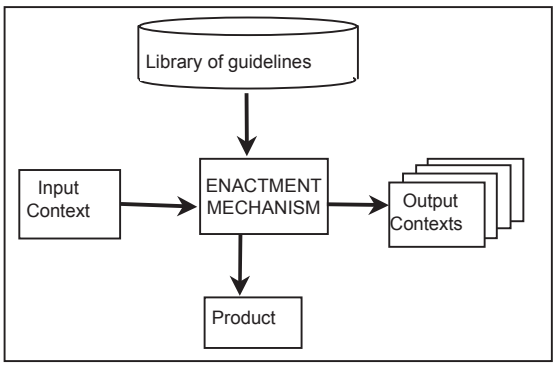

Figure 2: *The EKD decision making pattern*

The library contains EKD specific guidelines which are tailored to the way the EKD method suggests to work with the different EKD models (see section 2.1). Such guidelines suggest, for instance, different techniques for supporting the emergence of goals, the

operationalisation of goals, the classification of goals etc.. These guidelines are independent of any particular domain but are based on EKD method knowledge. Thus, if for example, the current context asks for the classification of a specific goal, the EKD guideline for goal classification can be applied.

Finally, if none of the two previous types of guidelines matches the current context of work, the generic guideline may operate. There is one single generic guideline. It is a generic rule for supporting decision making in change processes when neither EKD specific guidelines nor domain specific guidelines apply.

Section 3 will illustrate in more details the three types of guidance with the ATC case study. Taking into account the guidance focus of the decision making pattern, we can view the reasoning mechanism offered by the pattern as consisting of two main steps: selecting first, from the library the relevant guideline for the current situation and intention and, then making decision according to the guideline.

A guideline may be looked upon as a *strategy* for decision making in a given situation with an intention in mind. Applying the guideline provides the *tactics* for decision making. Enacting the tactics leads to *actions* that may change the product under development and generate new situations and new intentions, i.e. new contexts. Consequently, the decision making pattern can be viewed as suggesting a reasoning loop as shown in figure 3. Starting from an intention motivated by a given situation, a strategy must be selected, then converted into a tactics - which implements the strategy - leading to trigger the execution of actions generating new situations motivating new intentions.

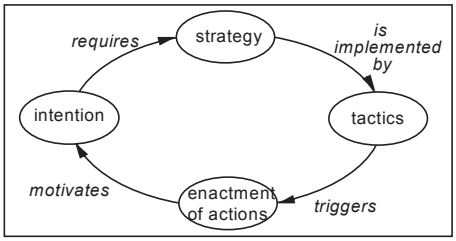

Figure 3: *The reasoning loop*

The reasoning loop will be exemplified in section 3, when we will illustrate the three types of guidance, together with the usage of this loop.

2.4. Clarifying the concepts used to describe the EKD decision making pattern

The EKD decision making pattern is defined at the type level of process modelling (see section 1). We present in this section the concepts (defined at the meta-type level) used to describe the EKD decision making pattern. During this presentation, we follow a bottom up approach starting with the concept of context, moving to the concept of product and its related sub-concepts and ending up with an overall view of the concepts progressively introduced. We will refer to this set of concepts as the *process meta-model* [12]. The presentation uses an extended binary entity-relationship notation [34], based on entity types, relationship types,

cardinalities, "is_a" links and objectified relationship types (an abstraction mechanism allowing us to view a relationship type as an entity type).

2.4.1. The concept of context

The key concept underlying the process modelling approach is the one of context. A *context* is defined as a pair <*situation*, *intention*> which couples a given situation and the intention one may wish to fulfil with regards to the situation, the goal. As a matter of fact, most of the time, it is the situation which motivates an intention. For example, it is because the "*Paris airport is loosing money*" (the current situation) that the goal of "*Increasing the number of passenger*" is raised. Similarly the "*availability of reliable bar-codes*" (the situation) suggests to "*Change the customer identification of baggage tags*" (the goal). In addition, a goal is not sufficient in itself as its interpretation and meaning may vary according to the situation it is associated with. The London airport might have the same goal of "*Reorganising the airport in order to increase passenger traffic*" but it will be achieved in a totally different way as the situation which raises this goal is that "*The airport can afford a larger number of passenger*". Acting in a context corresponds to a step in the EKD process.

Below are some examples of contexts which introduce our notations.

<"The Paris-Roissy Airport is loosing money", Reorganise the airport in order to increase passenger traffic>

<"The London Airport can afford a larger number of passenger", Reorganise the airport in order to increase passenger traffic>

<Goal G1 "Minimise risk of befalling aircrafts", Operationalise Goal G1 >

<Goal G1 "Minimise risk of befalling aircrafts", Define the type for Goal G1 >

We note that the same intention can be associated to different situations and conversely, the same situation can be associated to different intentions. The excerpt of the meta-model presented in figure 4 models the notion of context as an objectified relationship type. It is illustrated with the third examples.

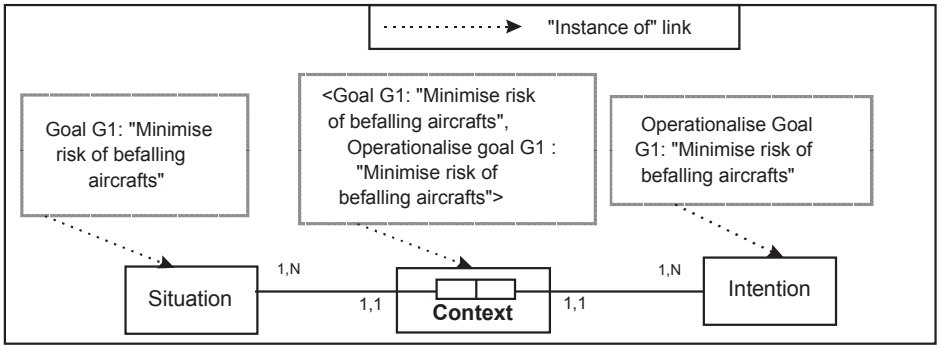

Figure 4: *An example of context*

Let us detail first the concept of *situation* which will lead us to introduce the related concept of product. Then, we will detail the concept of *intention*.

2.4.2. Situation, product and product parts

Whilst progressing during the change process, the EKD engineer focuses on different situations. As the EKD process is entirely based on artefacts (the so-called product), all situations which are referred to in the process are built over *product parts*. A *situation* comprises the set of product parts that are relevant for making the decision in the context in hand. For instance, a situation can be made of a single *goal* or a *set of activities*, a *hierarchy of actors,* etc.. all these being parts of the product. This leads to model the relationship *built over* between situation and product part as shown and exemplified in figure 5 and to define the product as *made of* product parts. The product describes the current and/or the desired states of the enterprise. It is modelled in terms of the concepts provided by the various EKD models, and therefore product parts are instances of the concepts of EKD models.

Situations can be expressed at different levels of granularity. It can be a single goal like in figure 5 or the entire product. We classify situations as *simple* or *complex*. A simple situation refers to a single element of the product (e.g. a goal) whereas a complex situation is composed of a set of product elements.

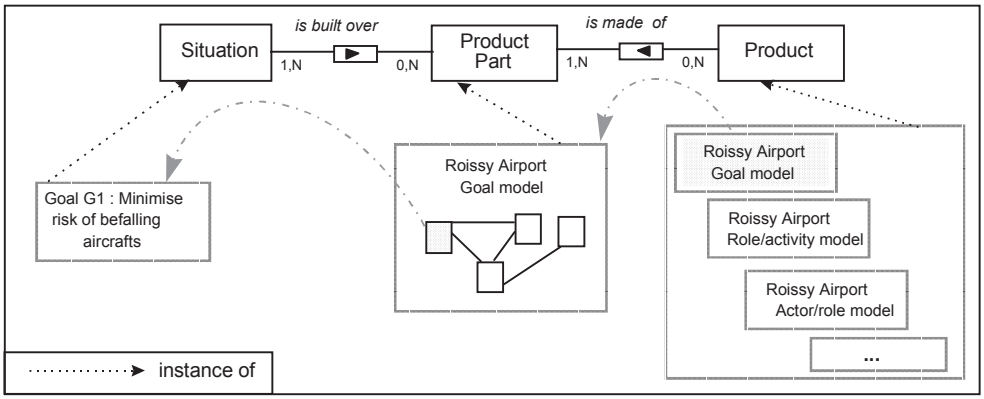

Figure 5: *Modelling situations with product part and product*

Figure 6 is the final view of the situation and product concepts as introduced at this stage.

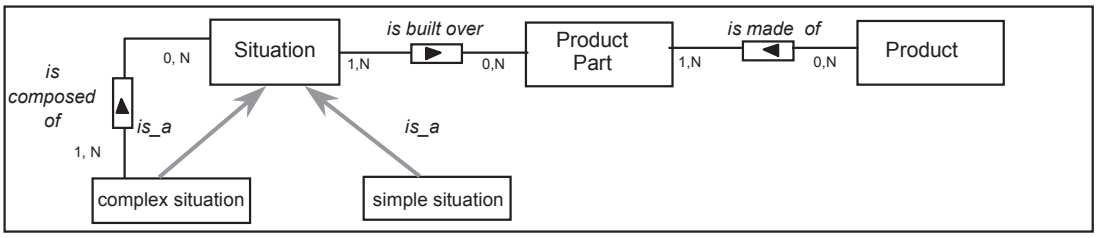

Figure 6: *The final view on the concepts of situation, product part and product*

2.4.3. Intention

When an EKD engineer considers a situation, he/she does it all the time with some purpose in mind, we say with some *intention*. An intention is a goal. Note that in this case the goal is not related to the organisation and therefore part of the EKD product but it is a goal of the EKD process. For instance, an engineer may want to operationalise a goal, to create a new goal, to set an issue, etc.. "*Operationalise goal*", "*Create a new goal*", "*Set issue*" are intentions.

An intention has always a target which describes the expected result of the fulfilment of the intention. Most of the time, the target is implicitly named in the intention. For instance, the target of "*Create a new goal*" is obviously the goal to be created. Targets are built over product parts. Having to explicitly describe the target of an intention may facilitate the understanding as well as the fulfilment of the intention. This leads to extend the meta-model as shown in figure 7.

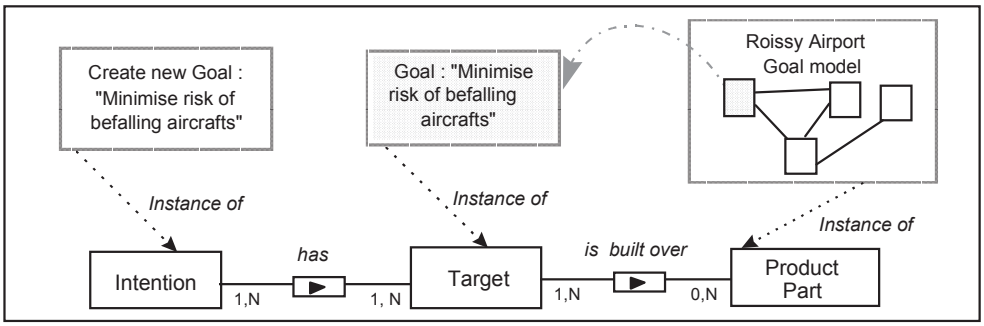

Figure 7: *Modelling the intention and its associated target*

2.4.4. Overview of the process meta-model

Figure 8 depicts the process meta-model as currently defined. It shows the emphasis on the concept of context being a pair <situation, intention>.

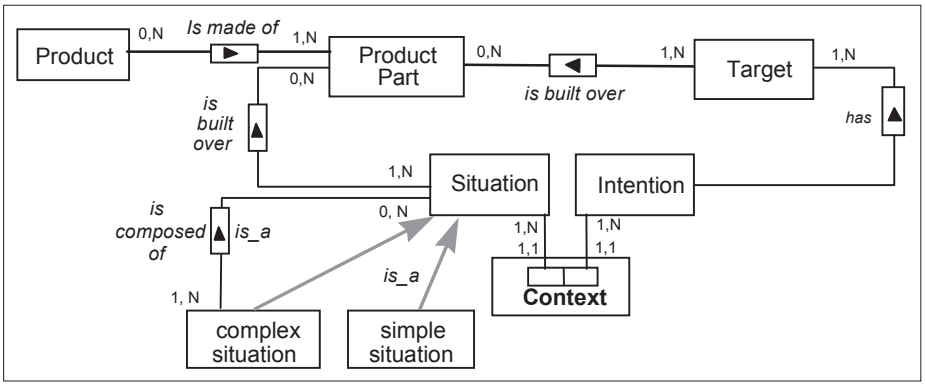

Figure 8: *Overview of the process meta-model*

3. Three types of guidance applying the decision making pattern

We shall describe in this section, how to work with these concepts using the EKD decision making pattern in order to get guidance and support in decision making. We shall focus in turn on the three types of guidance and illustrate the use of the pattern with the same example as an input: the context *<Goal G1: "Minimise risk of befalling aircrafts", Operationalise goal G1>*.

3. 1. *Generic guidance*

This section is devoted to the application of the decision making pattern, pattern for short, using generic guidance.

3.1.1. Using generic guidance

As mentioned before, the library of guidelines comprises only one generic guideline that we refer to as the *generic method guideline* or simply *generic guideline.* The guideline is applicable in *situations* where the two other types of guidelines do not hold for fulfilling the input contexts' intention, i.e. when there is no domain specific guideline available, nor EKD guideline meeting the current situation and intention. The guideline aims to fulfil the intention and is method independent. It proposes three options for progressing in the EKD process:

- The *do* option corresponds to a strait-forward resolution strategy. It should be chosen when the engineer knows exactly what needs to be done in order to fulfil the context's intention.
- The *choose* option corresponds to a resolution strategy which requires the exploration of alternative paths. It should be selected when the engineer thinks about different alternative ways for progressing with regard to the input context but has not make up his/her mind about the one to select.
- The *plan* option follows a planning strategy. The engineer has in mind a plan for achieving the context's intention. In this case, following a divide and conquer tactics, the EKD engineer will progress by building a plan of decisions to be made.

The two last options correspond to the classical reduction operator in the problem reduction approach to problem solving [23].

In the following, we exemplify the three options in turn with the example of the context *<Goal G1: "Minimise risk of befalling aircrafts", Operationalise goal G1>*, according to the steps of the reasoning loop introduced in section 2.3 (figure 3).

Case 1: The EKD engineer knows how to proceed to operationalise the goal G1 "*Minimise risk of befalling aircrafts*"

We can assume for example, that he/she has already worked on a similar problem or that a trace of a similar process is available and thus, he/she knows that the way to operationalise the goal is to reduce it into two other goals, G2: "*Maintain separation standards*" and G3: "*Decrease risk of human error*".

After having selected the *do strategy*, the EKD engineer is guided towards its implementation. The *do* strategy corresponds to a input context which we refer to as an *executable context*, i.e. a context whose intention is directly applicable through design *actions* implying changes on the *product*. The generic guideline encapsulates the *tactics* corresponding to this case. Therefore, the EKD engineer is asked to specify the action(s) to be performed on the product. In our example, the engineer describes (1) the creation of the two goals G2 and G3, and (2) the reduction structure expressing that G1 is decomposed into G2 and G3. This is shown in figure 9 together with the extension of the process meta-model related to the specialisation of *context* into *executable context*.

Enacting the defined tactics consists of executing the action(s) and leads to the modification of the product under development as shown on figure 9. The *do* strategy is associated to the characterisation of the input context as an executable *context*. A context is executable when its intention does not need any refinement and can be implemented immediately through actions on the product, i.e. in terms of modifications of the EKD models elements (creation of a goal, a rule, deletion of an element, etc.). Working with the *do* strategy means that the EKD engineer is able on the spot, to specify the actions which operationalise the process intention and to execute them immediately after. The process is constructed dynamically, the generic guideline helping in identifying the case in hand (the *do* strategy) and providing the procedure to be followed in this case (specifying actions and their impact on the product).

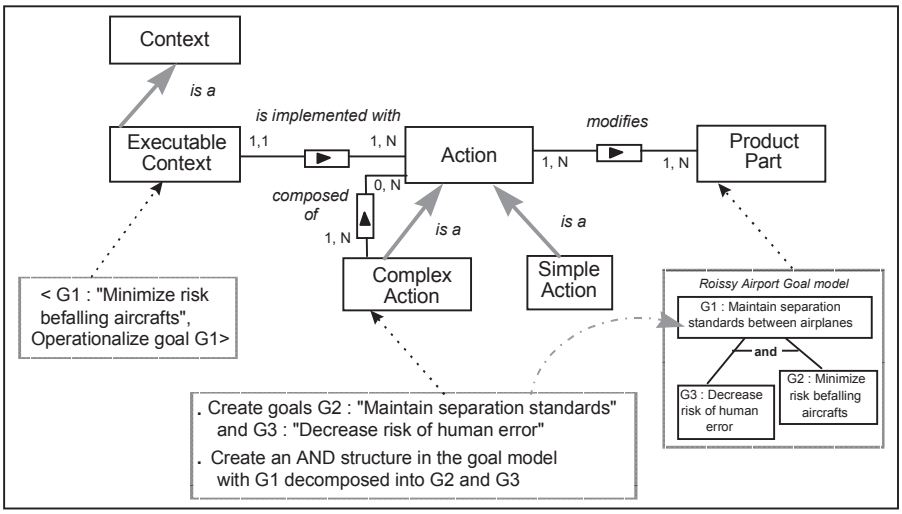

Figure 9: *Do strategy and executable context*

As shown in figure 9, an action can be either *simple* or *complex*. A simple action leads to modify a single element in the product whereas a complex action leads to changing several elements of the product and is composed of several simple or complex actions. In our

example, the action is complex because it is composed of three simple actions (creation of goal G2, creation of goal G3, and creation of the AND structure).

Case 2: The EKD engineer has choices in mind to operationalise the goal G1 "*Minimise risk of befalling aircrafts*".

The EKD engineer selects this *strategy* when he/she does not have a clear idea on how to deal with the input context, but has in mind different alternative ways of achieving its intention (the one of the input context). However, his/her analysis is not yet deep enough for deciding for the most appropriate one. The input context in this case is called a *choice context*. Assume in our example that while operationalising the goal "Minimise risk of befalling aircrafts", the engineer hesitates between (1) performing an AND reduction with two goals G2: *"Maintain separation standards"* and G3: *"Decrease risk of human errors"* and (2) performing an OR reduction with the same two goals G2 and G3. This is a case where the *choose* strategy proposed by the generic guideline should be selected.

In this case, guidance based on this strategy suggests the following *tactics*.

(1) to define the different alternative ways in terms of contexts. These are, in our example :
Alternative 1: < *G1: "Minimise risk befalling aircrafts", Perform AND reduction with G2: "Decrease risk of human errors" and G3: "Maintain separation standards"*>
Alternative 2: < *G1: "Minimise risk befalling aircrafts", Perform OR reduction with G2: "Decrease risk of human errors" and G3: "Maintain separation standards"*>

(2) to express pros and cons arguments for each alternative. These arguments will ground the final decision for the selection of the most appropriate alternative. In our example, the arguments could be :

Pro arguments for alternative 1 :
- Fulfilling both goals would reduce the company insurance fee
- G3 must be fulfilled to conform the state law N°1234
-...

Pro arguments for alternative 2 :
- The cost of fulfilling G2 and G3 is too high (according to Mr Smith's document)
- Fulfilling both goals G2 and G3 will make impossible to increase the traffic during summer.
...

The final decision is based on the arguments expressed to support or object to the identified alternatives. The output of the step in this case is a new context, the selected alternative context. Assume that the EKD engineer selects the AND reduction (i.e. alternative 1). This leads to generate the context < *G1: "Minimise risk befalling aircrafts", Perform AND decomposition with G2: "Decrease risk of human errors" and G3: "Maintain separation standards"*> as an output of the process step. In the following step where this context will be the input, the *do* strategy will be selected (in fact, this context is an executable one) and the

step will proceed similarly to what we exemplified for case 1. The end result of this second step will be a modification on the product leading to introduce an AND reduction for goal G1 with G2 and G3 as sub-goals (see figure 9).

Figure 10 graphically depicts the extension of the process meta-model required by the introduction of the new type of context, the *choice context* along with the instances describing our example. The thick black lines between the studied choice context and its alternatives is the graphical convention we used to represent the *alternative* relationship.

Again, in case 2, there exists a relationship between the strategy suggested by the generic guideline and the type of context, the *choose* strategy being applicable when the context is a *choice context.* Unlike the executable context (which application is strait forward), the choice context offers several choices, i.e. different ways to achieve the intention the EKD engineer has in mind at this precise point in time. As mentioned in case 1, the EKD process is here again dynamically constructed in a guided manner. Finally, we can notice that two turns in the reasoning loop are necessary in this case to fulfil the initial intention: a first one for refining the intention and a second one for implementing the refined intention by executing the consequent actions on the product.

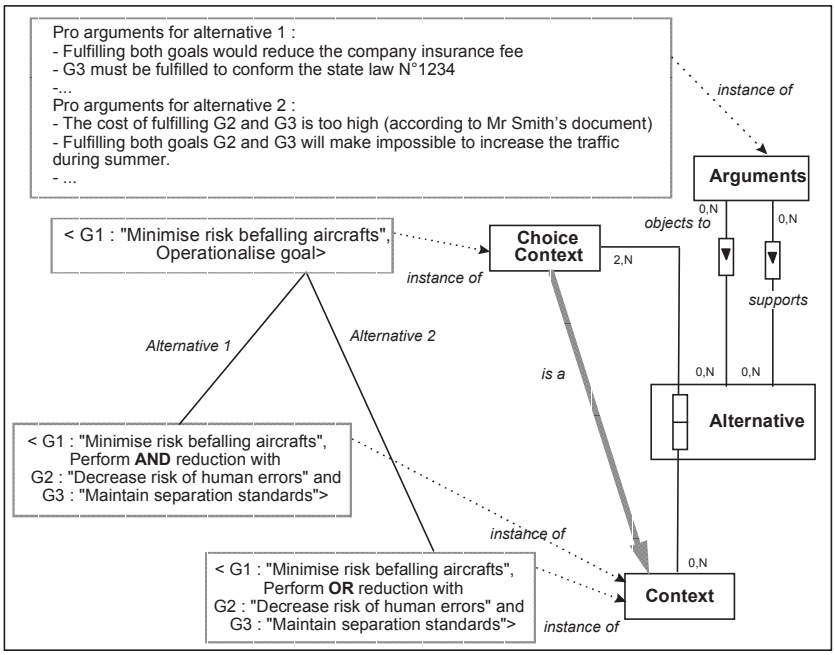

Figure 10: *Choose strategy and choice context*

Case 3: The EKD engineer realises that the fulfilment of the current context's intention requires the definition and the implementation of a plan of decisions.

The so-called *plan strategy* is selected when the achievement of the intention requires a composite decision making process but that is made up in the mind of the EKD engineer. In the example at hand, this case could correspond to a situation where the engineer has several hypotheses for reducing the goal "*Minimise risk befalling aircrafts*". He/she needs to set up and think about before being able to decide. His/her position is more confuse than in case 1 and even case 2, but he/she can built a plan for supporting his/her decision making. The type of

context associated to this strategy is called a *plan context*. In this case, several stages of decision making must be followed. Assume that the EKD engineer has no clue on how to operationalise the goal G1: "*Minimise risk befalling aircrafts*". The guidance provided by the plan strategy of the generic guideline suggests the following *tactics*:

(1) identify the different hypotheses he/she can think of, and express them using contexts. These are for our example:

< G1: "Minimise risk befalling aircrafts", Identify Hypothesis H1: "Maintain separation standards">

< G1: "Minimise risk befalling aircrafts", Identify Hypothesis H2: "Decrease risk of human error">

< G1: "Minimise risk befalling aircrafts", Identify Hypothesis H3: "Limit the number of aircrafts allowed to cross controlled airspace">

...

(2) express the arguments for and against each hypothesis. This is again expressed using contexts:

< G1: "Minimise risk befalling aircrafts", Express positive arguments for H1: "We have had accidents due to separation standard violations">

< G1: "Minimise risk befalling aircrafts", Express positive arguments for H2: "We have had accidents due to controller misjudgements">

< G1: "Minimise risk befalling aircrafts", Express positive arguments for H3: "We have had accidents due to heavy traffic in the controlled airspace">

< G1: "Minimise risk befalling aircrafts", Express negative arguments for H3: "Air traffic increased a lot during summer">

...

(3) define the appropriate goal reduction:

< G1: "Minimise risk befalling aircrafts", Define the appropriate reduction for G1 as G2: "Maintain separation standards" AND G3: "Decrease risk of human error">

The output of this reasoning loop is a context *(< G1: "Minimise risk befalling aircrafts", Define the appropriate reduction for G1 as G2: "Maintain separation standards" AND G3: "Decrease risk of human error">)* which is the input of the following loop. As this context is an executable one, its enactment will proceed as presented in case 1 or shall lead to change the product by creating the two goals G2 and G3 and relating them to G1 with AND reduction links.

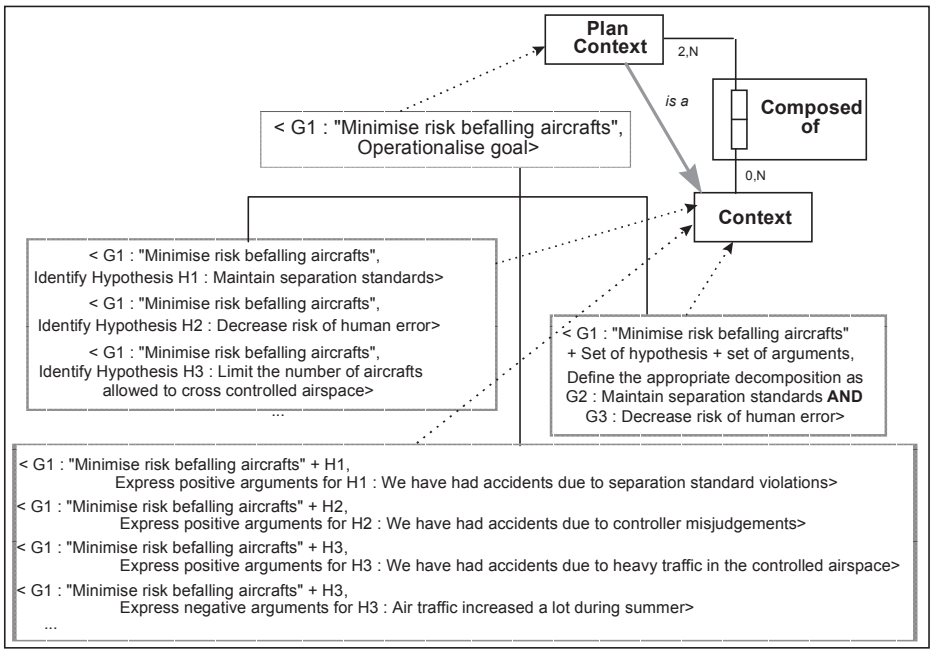

Figure 11: *Plan strategy and plan context*

Figure 11 graphically depicts the extension of the process meta-model required to support the *plan* strategy along with the instances describing our example. The rake is used to graphically represent the *"composed of"* relationship between the studied plan context and its components.

The plan context corresponds to a well-known approach in strategic decision making which is "*state a plan and then execute the plan*" [35]. Again, this is made here in a dynamic and guided manner. The EKD engineer working in a plan context will state the components of the plan he/she has in mind as new contexts and will enact them immediately after.

3.1.2. Remarks

First, it should be noted that the use of generic knowledge presented here corresponds to the weakest mode of guidance, it is the default mode and is suggested when there is no possibility of using neither EKD knowledge nor domain specific knowledge. In fact, the generic knowledge consists of using the process meta-model as a shell for structuring the way of progressing through the EKD process in a situation and intention oriented manner.

Second, we would like to stress the difference between the process and the product. We deliberately used the same example of context *<Goal G1: "Minimise risk of befalling aircrafts", Operationalise goal G1>* to demonstrate how the different strategies can be applied. In each case, the resulting product is identical, it consists in creating an AND reduction structure for the goal G1, with G2 and G3 as sub-goals. However, the process followed in each case is different and this can be recorded in the process trace.

In the first case, following the *do* strategy, there was no hesitation at all from the EKD engineer. In the second case, following the *choose* strategy, the process required two turns in the reasoning loop, one for deciding how to proceed and the second one to do. The *do* part is

the same in case 1 and in case 2 but the process trace in case 2 will record, in addition, the fact that the engineer envisioned two alternative ways of proceeding for the reduction of goal G1. The process trace keeps track, in this case, of the alternatives considered, the arguments expressed, and the alternative chosen. In the third case, following the *plan* strategy, the process trace shows the process that has been followed: the description of plan set by the engineer including the set of possible hypotheses he/she envisaged, the set of arguments and their relations with the hypotheses, and the selected reduction he/she came up with.

Process traces are different in each case even if the resulting product is identical. Keeping track of the process is useful for re-use in similar but different applications. In fact, *domain specific knowledge* will be extracted from process traces using, for instance, a case based learning approach [26].

Furthermore, the examples also bring out the analogy between the structure of the EKD process goals (we call them intentions for avoiding confusion) and the structure of the business goals in the product (described in the EKD goal model). In both cases, goals are reduced with either an AND or an OR structure or are directly implementable - through executable contexts and actions for the process and operationalised goals for the product.

3.1.3. Revising the process meta-model

The figure 12 summarises the different concepts completing the version of the process meta model we presented in figure 8. The new elements are in bold face.

The different types of contexts, namely *executable, choice* and *plan context*, are modelled as sub-types of the concept of context and therefore share the same structure <situation, intention>. The different components (respectively alternatives) of a plan context (respectively choice context) are contexts too. This provides the means to construct hierarchies of contexts which are necessary to represent complex decision based processes.

It may be argue that this process meta model is complex, or event too complex. However, the following principle is at the core of several machine learning system [17]: "*The more knowledge at the meta level, the more knowledge based guidance can be provided to acquire requirements fragments at the domain level."*

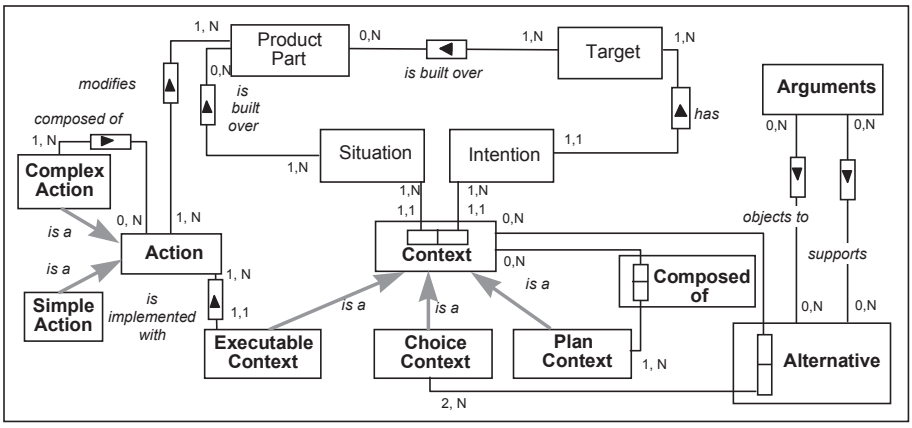

Figure 12: *The revised process meta-model*

3.1.4. Formalising the generic method guideline

As illustrated previously, the generic method guideline offers three different working strategies (to do, to choose and to plan) to help the EKD engineer when neither EKD nor specific guidance is applicable to fulfil an intention in a given situation. The reader understands that we advocate a formal representation of the generic guideline based on the process meta-model. In fact, our proposal is for a representation of all method guidelines in the librairy using the concepts of the process meta-model as defined now in figure 12.

Figure 13 visualises the *generic guideline* as a *choice context* using our graphical notations. The three strategies are the three choices proposed by the context together with their related arguments. The guidance provided by the generic guideline can be simply summarised by three questions to the EKD engineer :

- Is your intention operationalisable through design actions?
- Can you set alternative ways for fulfilling your intention?
- Do you need a plan for making your mind and achieve your intention?

The generic guideline has a fourth strategy which is dedicated to the modelling of co-operative processes and is not discussed in this paper: "*Brainstorm during a co-operative session*". This strategy led us also to perform several extensions on the meta-model. Those aspects were developed in [24].

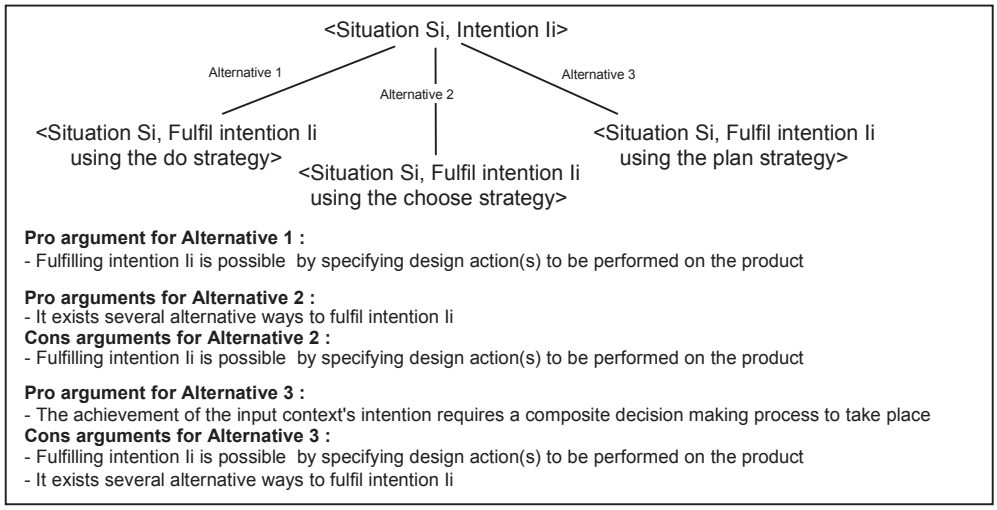

Figure 13: *Representing the generic knowledge with a choice context*

The alternative contexts shown in figure 13 are executable contexts whose associated actions guide the EKD engineer in doing things according to the option he/she chose.

3.2. EKD guidance

EKD guidance is based *on EKD knowledge,* i.e. knowledge specific to the EKD method for supporting EKD engineers to specifically undertake the change process in an organisation using the EKD models. Using this type of knowledge allows us to considerably speed up

EKD processes because it concentrates on the resolution of EKD specific problems. The quality of guidance is very high specially when supported by a tool environment [11].

3.2.1. Modelling EKD Knowledge

The EKD knowledge supports for example, the construction of the different models representing the initial enterprise state as well as the future enterprise state of the organisation, the expression of alternative strategies for change, the evaluation of these strategies, as well as other kinds of activity such as brainstorming, co-operative work, etc.. Section 4 introduces a wide variety of EKD guidelines and details some of them.

We express this knowledge as we did for the generic knowledge, i.e. using the process meta-model and the different types of context, namely executable, plan and choice contexts. However there is one major difference: the EKD knowledge is expressed at the type level i.e. the level of specific classes of EKD concepts such as "*Identifying goals*", "*Operationalising goals*", "*Finding design models meeting specific goals*" etc.. The type level is distinct from the instance level where one speaks of a specific goal or of a specific design model. It has also, to be differentiated from the meta-level dealing with generic concepts such as "product part" and "intention". The so-called generic knowledge is at the meta-level whereas the EKD knowledge is at the type level.

As an example, let us assume that the reasoning followed for operationalising the goal "*Minimise risk of befalling aircraft*" - where hypotheses and arguments have been expressed first, followed by the definition of the appropriate reduction - was observed in similar forms in different change processes. The experience gained by the EKD engineers suggests a generalisation of the plan followed (depicted in figure 11, section 3.1.1) into an EKD guideline expressed at the type level with a plan context having a goal G as situation and "*Operationalise goal G*" as intention. Associated to this plan context is a set of component contexts for describing the different decisions to be made in order to fulfil the intention "*Operationalise goal G*". The guideline is expressed at the type level as it does not deal with a specific situation (a specific company goal) and a specific intention, but a class of situations (the class of company goals) and a class of intentions (the set of "*Operationalise goal*" intentions). As it is expressed at the type level, this plan can be re-used in many contexts: all the contexts dealing with the operationalisation of a goal. Figure 14 graphically depicts the guideline obtained by generalisation of the plan stated in case 3, section 3.1.1.

The bottom of figure 14 depicts also what we call a *dependency graph*. This graph should be defined for each plan context in the EKD method base. It provides information about the various possible ordering paths for decision making within the plan. This information is helpful for providing more advanced guidance as it offers more flexibility in the process of decision making than the usual sequential ordering of steps. The dependency graph permits to describe iteration, backtrack as well as parallelism. Concerning the operationalisation of a goal, the graph suggests to iterate on hypotheses identification until all of them have been identified (argument A1), then to iterate on argument expression (argument not A3). Note that

at this point it is possible to resume hypothesis identification and to go back and forth between the two contexts. The plan ends with the definition of the appropriate decomposition of goal G.

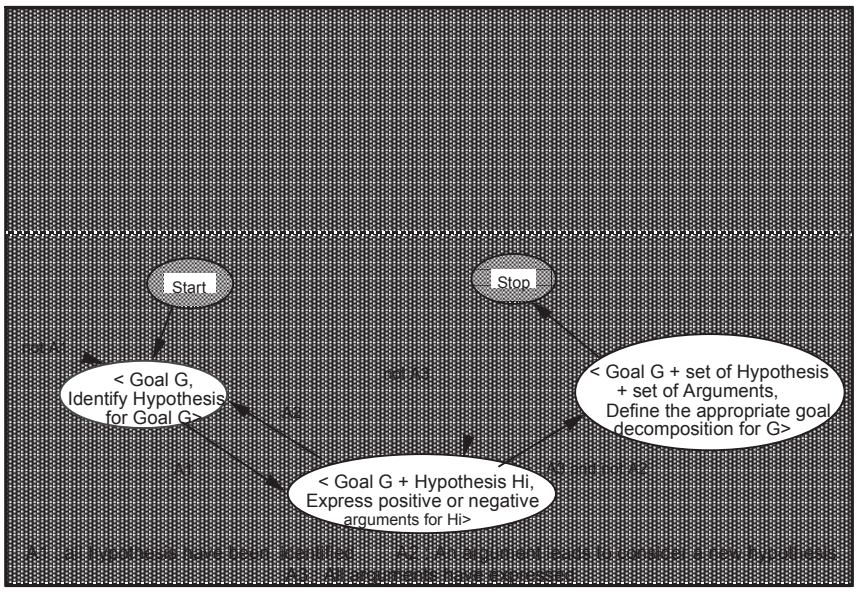

Figure 14: *An EKD guideline for operationalising a goal*

The EKD knowledge can be re-used for decision based guidance in many different change processes within different companies. An EKD guideline is re-usable any time the situation type matches elements of a specific design product and the intention type matches the current intention of the EKD engineer.

3.2.2. *Using EKD guidance*

The way EKD knowledge is used within the EKD decision pattern is similar to what we have illustrated for the generic guidance. The main difference lies in the retrieval of the guideline. The retrieval of an EKD guideline is based on pattern matching. Assuming that the engineer has chosen the input context, he/she has to select an EKD guideline where (1) the situation type matches the input context's situation and (2) the intention of the guideline matches the input context's intention. This selection is facilitated by the use of a software tool [11].

The remaining part of the reasoning loop associated to the application of the EKD decision making pattern is similar to what we presented for using the generic knowledge but the EKD engineer is more guided:

- the tactics is provided by the EKD guideline. However, in this case, the EKD engineer is only required to instantiate the guideline. He/she does not have to find himself/herself the way of resolving the issue he/she is faced to, but he/she is just required to follow the predefined resolution approach provided by the guideline. If the guideline is a choice context, the EKD engineer will have to instantiate the alternative contexts whereas he/she will do the same for the component contexts of a plan; he/she does not have any thing to do at this stage if the context is executable.

- the enactment is identical to what we presented earlier. The EKD engineer makes decisions according to the pre-defined tactics, i.e. selecting the most appropriate alternative based on the arguments provided by the guideline in case of a choice context; selecting the adequate path in the precedence graph to execute a plan context and performing the action(s) of an executable context to modify the design product accordingly to the decision made.

Let us demonstrate the use of EKD guidance in the case of our example with the context *<Goal G1 "Minimise risk befalling aircrafts", Operationalise Goal G1>*. The context matches the situation and decision of the guideline depicted in figure 14. The instantiation of the predefined component contexts is illustrated in figure 15. The enactment is identical to what was presented in section 3.1.1 (case 3). In fact, the guideline shown in figure 15 was obtained by generalising the plan context that has been previously constructed using the generic guidance.

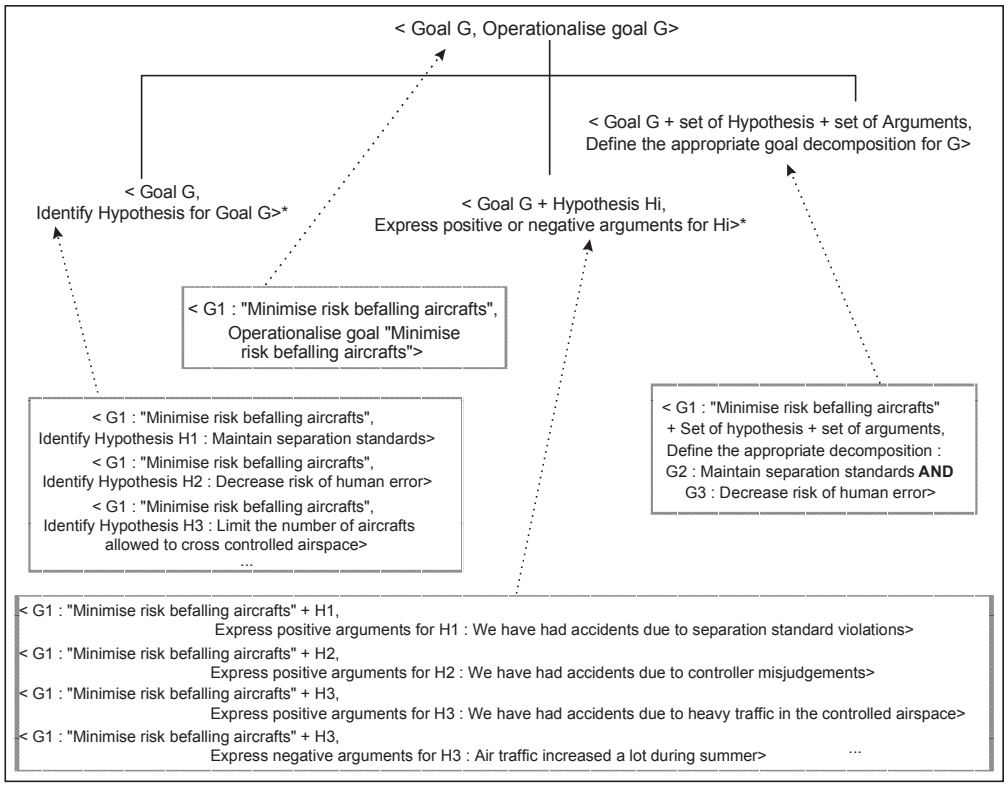

Figure 15: *Instantiating the EKD guideline depicted in figure 3.14*

By following the heuristical knowledge embedded in the EKD guideline, the engineer is constantly guided. Part of the solution he/she has to find is provided by the guideline. Suggestions are made on the alternative strategies he/she can follow, pre-defined arguments supporting or objecting to these strategies are provided, predefined plans he/she can use for reaching his/her decision are ready-made, he/she is told what to do next, etc..

3.3. *Domain specific guidance*

Domain specific guidance is based on *domain specific knowledge*. The domain specific knowledge aims at providing guidance to EKD engineers for solving very well focused problems related to a specific domain. It is grounded on experience based knowledge and

suggests to re-use and to adapt concrete and already tested solutions which have shown their efficiency and feasibility in different organisational settings of the same domain. The benefits of using this type of knowledge is that it considerably shortens the decision process of solving domain specific problems by offering ready to use solutions.

3.3.1. *Modelling domain specific knowledge*

For instance, in the context of air traffic control, domain specific knowledge could suggest that the operationalisation of the goal G1: "*Minimise risk of befalling aircraft*" can be achieved in two alternative ways. It can be either a decomposition of G1 into two complementary goals G2: "*Maintain separation standards*" and G3: "*Decrease risk of human error*" or a decomposition with G3 and G4: "*Limit the number of aircrafts allowed to cross controlled airspace*".

Similarly to generic and EKD knowledge, domain specific knowledge is captured in guidelines expressed as hierarchies of contexts of the three different types. However, domain specific knowledge is described at the instance level. It deals with process intentions related to concrete facts such as the intention of operationalising the airport goal of "*Minimising the risk of befalling aircraft*" or the intention of modelling the design solution based on "*the identification of customers' baggage using bar-codes*".

Figure 16 depicts the guideline related to the operationalisation of the goal G1 as introduced above. In addition to the two alternative contexts describing the two ways of operationalising the goal "*Minimise risk of befalling aircrafts*", an argument is provided (A1) to help the selection of the most appropriate alternative.

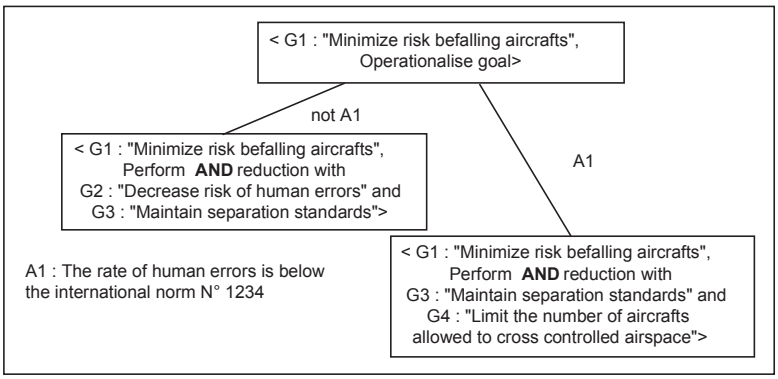

Figure 16: *An example of a domain specific guideline*

3.3.2. Using domain specific guidance

The use of domain specific knowledge within the EKD decision making pattern is very close to what has been presented in 3.2 for EKD guidance. The difference lies in the fact that domain specific guidelines are defined at the instance level and therefore do not have to be instanciated while being used.

The reasoning loop starts with the retrieval of the domain specific guideline matching the input context. This can be made manually by the EKD engineer who browses the library and looks for a guidelines where the situation is the input context's situation and the intention is

the input context's intention. It is more easily made with the use of the EKD tool environment which could realise the matching automatically [11], this has not been implemented yet. If there exists a guideline that matches, the EKD engineer can decide to use it. The guideline provides tactics for fulfilling the intention. The engineer has just to implement the them.

Let us reconsider our example with the input context *<Goal G1 "Minimise risk of befalling aircrafts", Operationalise Goal G1>*. Because this context is the same than the top context of the domain specific guideline depicted in figure 16, this guideline can be selected by the engineer. Enacting the guideline consists in studying the two suggested alternatives, evaluating the argument - that is to find out if "the rate of human errors is bellow the international norm N°1234" and then, selecting the alternative he/she believes to be the most appropriate. Assume the airport has a high rate of human errors, therefore the engineer chooses the alternative suggesting to perform an AND reduction on goal G1 with the two goals G2: "*Decrease risk of human errors*" and G3: "*Maintain separation standards*". This alternative is an executable context and its associated action precisely describes how to modify the design product in order to get the adequate reduction for goal G1 in the product.

4. EKD guidelines overview

As presented in section 3, guidelines are modelled with our process meta-model in terms of *contexts* and *trees of contexts*. Besides they are clustered in three classes: (1) The generic guideline, (2) EKD guidelines, (3) Domain specific guidelines.

As highlighted in section 3, each class of guideline is used by the decision making pattern to provide a specific type of guidance (generic guidance, EKD guidance and domain specific guidance, respectively). This section aims (1) to present a brief overview of the EKD guidelines and (2) to demonstrate the use of some of them.

4.1. *The EKD guidelines matrix*

We use a *matrix presentation* to overview the collection of EKD guidelines. These guidelines are the *matrix elements*. The *columns* of the matrix are *intentions* which arise during the EKD process. "*Operationalise goal*", "*Classify goal*" "*Identify goal*" are examples of such intentions. For the sake of readability, we use only intention names whereas of the full context heading, i.e. a situation name + an intention name, is really meant. 5 main intentions are identified: (1) Model the current enterprise state, (2) Acquire goal, (3) Operationalise goal, (4) Generate design models, (5) Validate design models. Some of them are decomposed in a number of sub-intentions which are visualised in the matrix as sub-columns. The *rows* of the matrix are *techniques* used in the guidelines. The same technique can be used in different ways in different guidelines. For example, SWOT analysis is a technique which might be used for satisfycing the intention of "*Analyse the context of change*" and for "*Argument reduction* (of a goal)" as well. The matrix of figure 17 summarises how techniques are embedded in EKD guidelines to support the various EKD process intentions.

As we have seen in section 3.2.1, the EKD guidelines presented in the matrix are domain independent. They can support the construction of the EKD Models for a given company, the expression of alternative strategies for change, brainstorming activities, etc..

The knowledge embedded in the EKD guidelines is expressed using the process meta-model developed in section 3.1. These guidelines can be re-used for decision based guidance in different change processes within different companies.

Context / Technique	(1) Model the current enterprise state	(2) Acquire goal								(3) Operationalise goal			(4) Generate design model			(5) Validate design model	
		2.1 Analyse the context of change	2.2 Find Goals	2.3 Situate Goal	2.4 Classify Goals	2.5 Prioritise goal	2.6 Detect goal conflict	2.7 Solve goal conflict	2.8 Relate goal	3.1 Reduce goal	3.2 Argument reduction	3.3 Identify business rule	4.1 Identify design model	4.2 Detail design model	4.3 Argument alternative design models	5.1 Evaluate design	5.2 Select design options
EM construction strategy	1			5					12								
SWOT analysis		2									15				22		
Categorizer			3					10									30
Goal typology					6												
Pair-Wise comparison						7											
Theorem proof							8										
Do/Choose/Plan strategies										13	16	18	19		23		
Goal template										14							
Brainstorming strategy			4				9	11			17				24		
Graph construction strategy														20			
View models														21			
Scenario																25	
Argument based reasoning																26	
Process trace																27	
Cost /resource evaluation																28	
Agent Dependency model																29	31

Figure 17: *The EKD guidelines*

4.2. Illustrating the set of guidelines supporting the "Classify goal" intention

This section presents the set of guidelines provided for classifying goals. Goal classification occurs after goal acquisition. The aim of the classification is to produce some partial views of the problem, each category focusing on different aspects of change and thus conforming to different needs. By doing so, the different stakeholders are able to concentrate on different facets of change separately and to improve their overall understanding of the problem, this leading to the possible identification of new goals. Goal classification leads to organise goals into coherent clusters.

Four guidelines are available for the intention "*Classify goal*". The first one does not suggest to use any pre-defined category [19] whereas the two others are based on existing categories proposed in the literature [5], [1], and are relevant for classifying goal in the

context of supporting the change process in an organisation. We present these guidelines in turn and illustrate how they provide guidance through some examples.

4.2.1. Classifying goals with user defined categories

According to [19], goal classification based on user defined categories is relevant for handling changes and for structuring a complex domain with users defined views. Guiding such a classification is supported by the guideline depicted in figure 18. The top context is: *<(Goal G + a set of user defined categories), classify G>*. The situation comprises a goal G and a set (possibly empty) of already defined categories. The guideline suggests to follow a plan having two components: (1) to start by defining the appropriate category and (2) to place the goal in the category. However, as shown on the dependency graph, it is possible to directly place the goal being studied in an existing category (if A3 is true). Furthermore, after having categorised a goal, it may be necessary to re-organise categories if one category contains too many goals (if A2 is true). In this case, the engineer is guided to define sub-categories and then re-organising the classification scheme. Finally, a goal can belong to several categories, the EKD engineer is guided towards multiple classification thanks to the loop put on context *< (Goal G + a set of categories C), Place G in C>* (if A4 is true).

Let us consider that the goal acquisition phase has led to the emergence of the following list of goals: (G1) Improve air-traffic flow, (G2) Reduce demands placed on controllers, (G3) Minimise risk of befalling aircrafts, (G4) Reduce delays, (G5) Reduce noise, (G6) Reduce fuel consumption.

Assuming that no category has been defined so far, classifying goal G1 "*Improve air-traffic flow*" would be guided as follows. First, category definition is asked (because A3 is false), this leads to the identification of the "Efficiency" category. Then, the guideline suggests to place G1 in a category, i.e. "Efficiency". Similarly, goal G2 is placed in two new categories called "Controller" and "Safety". The classification of goal G3 leads to directly place it in the existing "Safety" category. Similarly, goals G4 and G5 are put in the existing category "Efficiency". While classifying goal G6, the EKD engineer is suggested to refine the category "Efficiency" and is guided towards the identification of more specialised categories, such as "Cost efficiency", "Noise efficiency", etc..

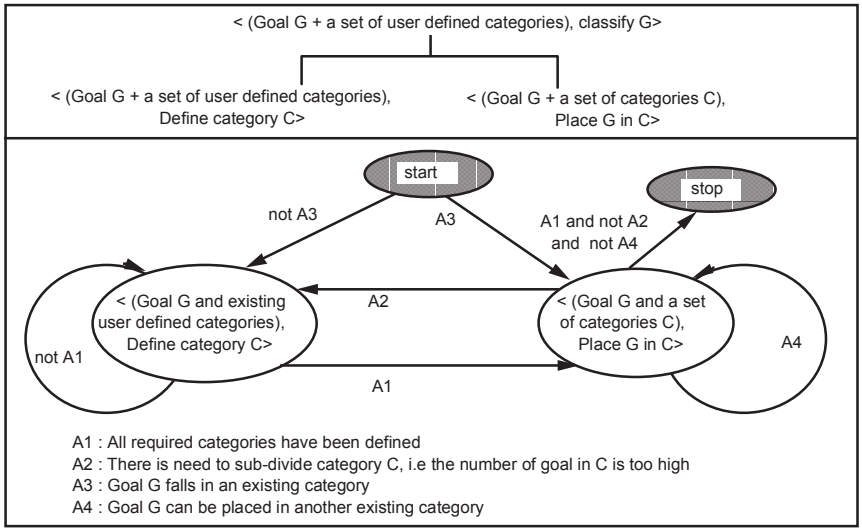

Figure 18: *The guideline for classifying a goal using user defined categories*

4.2.2. Classifying goals following a behaviour oriented classification

The following guideline is based of the classification proposed in [5]. This classification suggests to clusters goals according to the impacts they have on the set of possible behaviours of the system. Five categories are advocated: Achieve, Cease, Maintain, Avoid and Optimise. Achieve and Maintain generate behaviours, Cease and Avoid restrict behaviours, whereas Optimise compares behaviours.

The guideline suggests to study five alternatives for classifying a goal with regards to its impacts of the system behaviour. The different arguments provided for each alternative shall be carefully studied. Indeed, arguments are provided to help in the selection of the most appropriate one. Each alternative context is an executable context, its associated action adds the corresponding category to the goal description.

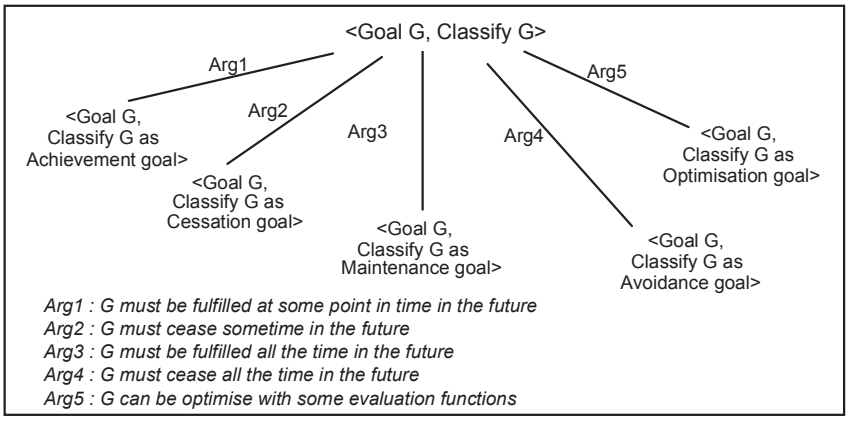

Figure 19: *The guideline for behaviour oriented classification*

Let us apply this guideline to the goal "*No passenger should miss his/her correspondence*". Let us assume that, after a careful study of the different arguments, the EKD engineer decides that the argument 3 applies, i.e. "this goal must be fulfilled all the time in the future. The enactment of the corresponding alternative adds the category "Maintenance goal" to the description of the goal *"No passenger should miss his/her correspondence"*.

4.2.3. Classifying goals into private goals and different types of system goals

This guideline advocates to distinguish between private goals and system goals and to refine systems goals. It is based on the classification proposed in [5]. Private goals are goals that *might* be achieved by the system. System goals are application specific goals that *must* be achieved by the system. Furthermore, several sub-categories have been defined for system goals, namely, *Satisfaction goals, Information goals, Robustness goals, Consistency goals, Safety goals* and *Privacy goals*. Satisfaction goals are concerned with satisfying agent requests. Information goals are concerned with getting agents informed about object states. Robustness goals are concerned with recovering from foibles of human agents or from breakdowns of automated agents. Consistency goals are concerned with maintaining the consistency between the automated and physical part of the system. Safety goals and Privacy goals are concerned with maintaining agents in states which are safe and observable under restricted conditions respectively.

The guideline embedding such a knowledge is depicted in figure 20. The top level context is a choice context with two alternatives, each alternative corresponding to respectively private and system goals. The context <*Goal G, Classify G as system goal*> is a choice context. It has six alternatives, each one related to one of the categories we just mentioned. Indeed, arguments are provided for each alternative, they are based on the definition of each goal category. All the other contexts depicted in figure 20 are executable contexts, their actions add a category to the goal described in the situation.

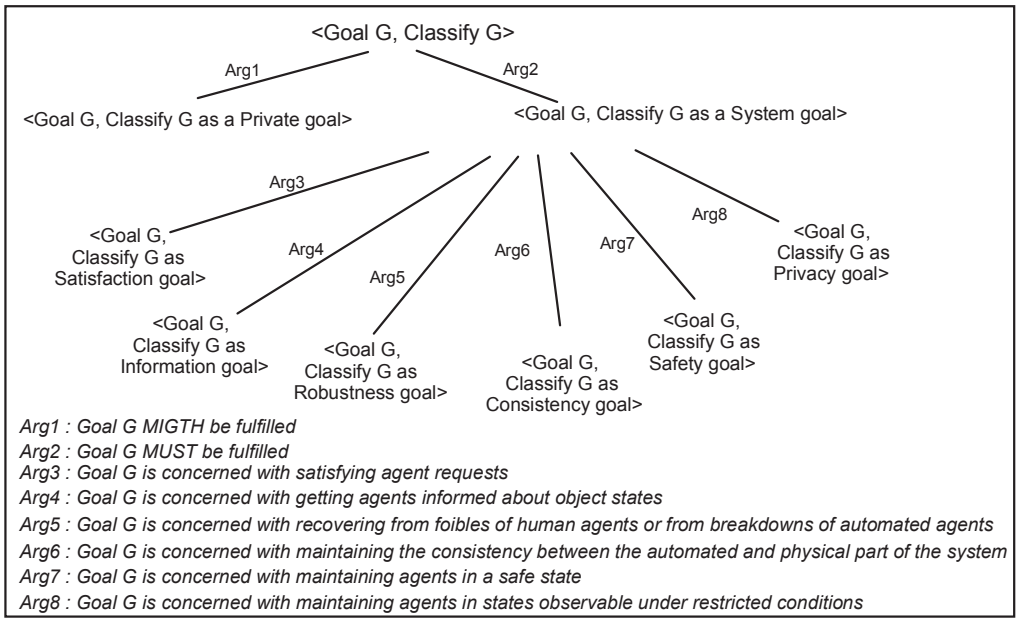

Figure 20: *The guideline differentiating private and system goals and refining system goals*

Let us take two examples showing how guidance can be provided by this guideline. We first consider the goal "*Maintain safe transportation*". The guideline suggests to decide if the goal must or might be fulfilled. Evidently, this goal must be fulfilled. Therefore the alternative <*Goal "Maintain safe transportation", Classify as System goal*> is selected. Then,

the guideline suggests to consider the arguments of the six alternatives and to decide on those applicable. In our example, it is clear that argument 7 applies (Goal G is concerned with maintaining agents in a safe state. Therefore, the context *<Goal "Maintain safe transportation", Classify as Safety goal>* is selected. As this context is an executable context, its enactment leads to automatically add two categories to the goal "*Maintain safe transportation*": "System" and "Safety".

Let us now consider the goal "*Avoid plane departure delay*". As a matter of fact, plane departure delay may be due to several unpredictable and external causes (the plane is not ready for taking off, the captain is not arrived on time, etc.) and therefore its fulfilment cannot be insured. Having taken into account these consideration, the EKD engineer decides to select the first alternative of the top choice context: the context *<Goal "Avoid plane departure delay", Classify as Private goal>*. As this context is an executable context, its enactment leads to add the "Private" category to the goal "*Avoid plane departure delay*".

5. Conclusion

This paper advocates for effective guidance to conduct enterprise knowledge modelling and enterprise change management. To this end, we propose :

- An intention based framework for describing guidelines for constructing enterprise models expressed in a process meta-model,
- A repository of guidelines described as method guidelines at three levels of abstraction : domain specific guidelines, EKD guidelines and a generic guideline that shall be used when no other type of guideline is applicable,
- A decision making pattern using the repository of guidelines and an enactment mechanism,
- A tool implementing the decision making pattern comprising EKD guidelines [11],
- An electronic guidebook[2] providing a set of guidelines to understand the EKD method and to guide stakeholders involved in the change process to manage it [25].

The process meta-model allows us to deal with many different situations in a flexible, decision-oriented manner [28], [24]. Moreover, it can support different levels of granularity in decision making as well as non-determinism in process performance. It identifies a decision in context as the basic building block of process models and permits their grouping into meaningful modules. Parallelism of decisions and ordering constraints are also supported.

The decision making pattern is tailored to provide guidance in all cases. In some cases, the pattern offers a domain specific guidance. This happens when the library contains knowledge about the domain of the project which matches the current context of work. The library contains EKD specific guidelines which are tailored to the way the EKD method

[2] http://www.univ-paris1.fr/CRINFO/EKD-CMMRoadMap

suggests to work with the different EKD models. These guidelines are independent of any particular domain but are based on EKD method knowledge. Finally, if none of the two previous types of guidelines matches the current context of work, the generic guideline may operate. It is the default option in some sense. Clearly, making guidance more specific increases its efficiency. However, the generic guideline, by offering a general frame for decision making, makes the EKD process entirely based on guidance.

Taking into account the guidance focus of the decision making pattern, we can view the reasoning mechanism offered by the pattern as consisting of two main steps: selecting first, from the library the relevant guideline for the current situation and intention and, then making decision according to the guideline.

To sum-up, the EKD process model suggests an incremental production of the enterprise knowledge description. It has two major advantages: it makes change traceable and it helps stakeholders in the change process to share awareness by making the product under construction being discussed, visible and explicit.

The result of this work has been effectively used in the context of the ELEKTRA project (ESPRIT project N° 22927) for handling the change in electricity companies due to the deregulation rules set by the E.C. The EKD guidelines have been used to produce part of the models describing both the current state and some future states of two electricity companies (see [22] for details). Furthermore, in this context, domain specific guidelines have been expressed following the proposed framework as change process patterns [27], [32].

References

[1] A. Anton, Goal-Based Requirements Analysis, in : Proceedings of ICRE '96, IEEE, Colorado Springs, Colorado USA, 1996, 136-144.

[2] J. Bubenko, Enterprise Modelling, Ingénierie des Systèmes d'Information 2 (6) (1994).

[3] J. Bubenko, J. Stirna, EKD User Guide, February 1997, ELEKTRA internal report.

[4] P. Checkland, J. Scholes, Soft Systems Methodology in action, John Wiley and Sons, 1990.

[5] A. Dardenne, A.v. Lamsweerde, S. Fickas, Goal-directed Requirements Acquisition, Science of Computer Programming, Vol. 20 (1993) 3-50.

[6] S. Decker, M. Daniel, M. Erdmann, R. Studer, An enterprise reference scheme for integrating Model based knowledge engineering and enterprise modelling, in : Proceedings of the 10th European Workshop on Knowledge Acquisition, Modeling and Management, EKAW'97, Lecture Notes in Artificial Intelligence, Springer-Verlag, Heidelberg, 1997.

[7] M. Dowson, C. Fernstrom, Towards requirements for Enactement Mechanisms, in : Proceedings of the European Workshop on Software Process Technology, 1994.

[8] R. Espejo, R. Harnden R. (eds). The Viable System Model : Interpretations and Applications of Stafford Beer's VSM, Wiley, Chichester, 1989.

[9] R.L. Flood, M.C. Jackson, Creative Problem Solving. Total System Intervention, John Wiley and Sons Ltd, 1991.

[10] V. Gilgeuos, Operations and the Management of Change, Pitman Publishing, 1997.

[11] C. Gnaho, J. Barrios, A Web based tool for helping IS engineering in method use, in : Proceedings of the Second International Workshop on Many Facets of Process Engineering, Gammarth, Tunisia, May 1999.

[12] G. Grosz, C. Rolland, S. Schwer, C. Souveyet, V. Plihon , S. Si-Said, C. Ben Achour, C. Gnaho, Modelling and engineering the requirements engineering process : an overview of the NATURE approach, Requirements Engineering Journal, N° 2 (1997), 115-131.

[13] K. Hung, T. Simons, T. Rose, The Truth Is Out There ? : a survey of Business Objects, in : Proceedings of the International Conference on Object oriented Information Systems, Paris, France, September 1998.

[14] S. Jarzabek, T.W. Ling, Model-based support for business reengineering, Information and Software Technology, N° 38 (1996) 355-374.

[15] V. Kavakli, P. Loucopoulos, Goal-driven business process analysis - Application in electricity deregulation , in : Proceedings of 10th International Conference CAISE'98, Pisa, Italy, June 1998, 305-324.

[16] J.P. Kotter, Leading Change : why transformation efforts fail, IEEE Engineering Management Review, Spring 1997.

[17] A.v. Lamsweerde, Learning Machine Learning, Introducing a Logic Based Approach to Artificial Intelligence, Vol 3, Wiley, 1991, 263-356.

[18] K. Lewin K, Group Decision and Social Change, Readings in Social Psychology, edited by E. Maccoby, T. Newcomb, E.H. Hartley , Rinehart & Wisnston, New York, 1958.

[19] P. Loucopoulos, P. Louridas, V. Kavakli, A. Tzanaki, D. Fillipidou, Contribution to the EKRD Approach, ELEKTRA Research Report, UMIST, Manchester, UK, October 1996.

[20] P. Loucopoulos, V. Kavakli, N. Prekas, C. Rolland, G. Grosz, S. Nurcan, Using the EKD approach : The modelling component, ELEKTRA Research Report, April 1997.

[21] P. Loucopoulos, V. Kavakli, N. Prekas, C. Rolland, G. Grosz, S. Nurcan, PPC Distribution Models, ELEKTRA Athena Deliverable, July 1997.

[22] P. Loucopoulos, V. Kavakli, N. Prekas, I. Dimitromanolaki, N. Yilmazturk, C. Rolland, G. Grosz, S. Nurcan, D. Beis, G. Vgontzas, System design specification for PPC, ELEKTRA Demetra Deliverable, March 1998.

[23] N. Nilsson, Problem Soving Method in Artificial Intelligence, McGrawHill, 1971.

[24] S. Nurcan, C. Rolland, Meta-modelling for cooperative processes, in : Proceedings of the 7th European-Japanese Conference on Information Modelling and Knowledge Bases, Toulouse, France, May 1997, 297-315.

[25] S. Nurcan, C. Rolland, Using EKD-CMM electronic guide book for managing change in organisations, in : Proceedings of the 9th European-Japanese Conference on Information Modelling and Knowledge Bases, Iwate, Japan, May 1999, 105-123.

[26] N. Prat, Using Machine Learning Techniques to Improve Informations Systems Development Methods, in : Proceedings of Americas Conference on Information Systems, Phoenix,USA, August 1996.

[27] N. Prekas, P. Loucopoulos, C. Rolland, G. Grosz, F. Semmak, Using patterns as a mechanism for assisting the management of knowledge in the context of conducting organisational change, in : Proceedings of the 10th International Workshop on Database and Expert Systems Applications, DEXA'99, Florence, Italy, September 1999.

[28] C. Rolland, C. Souveyet, M. Moreno, An approach for defining ways-of-working, Information Systems Journal 20 (4) (1995).

[29] C. Rolland, Understanding and Guiding Requirements Engineering Processes, invited talk, IFIP World Congress, Camberra, Australia, 1996.

[30] C. Rolland, S. Nurcan, G. Grosz, Guiding the participative design proces, Association for Information Systems Americas Conference, Indianapolis, Indiana, USA, August 1997, 922-924.

[31] C. Rolland, P. Loucopoulos, G.Grosz, S. Nurcan, A framework for generic patterns dedicated to the management of change in the electricity supply industry, in : Proceedings of the 9th International Workshop on Database and Expert Systems Applications, DEXA'98, Vienna, Austria, August 1998, 907-912.

[32] C. Rolland, G. Grosz, S. Nurcan, W. Yue, C. Gnaho, An electronic handbook for accessing domain specific generic patterns, in : Proceedings if the IFIP WG 8.1 Working Conference : Information Systems in the WWW environment, Beijing, Chine, July 1998, 89-111.

[33] S. Si-Said, C. Rolland, G. Grosz, MENTOR: A Computer Aided Requirements Engineering Environment, in : Proceedings of the 8th CAISE Conference on Challenges In Modern Information Systems, Heraklion, Crete, Greece, May 1996.

[34] TEMPORA Esprit project, Final report, 1994.

[35] Wilenski, Planning and Understanding , Addison Wesley, 1983.

[36] J. D. Wynekoop, N. L. Russo, System Development methodologies: unanswered questions and the research-practice gap, in : Proceedings of the 14th ICIS Conference (eds. J. I. DeGross, R. P. Bostrom, D. Robey), Orlando, USA, 1993, 181-190.

[37] ELectrical Enterprise Knowledge for TRansforming Applications, The ELEKTRA Project Programme, ELEKTRA consortium (1996).

FIGURE LEGENDS