Requirements Eng (1999) 4:169–187
© 1999 Springer-Verlag London Limited

Requirements
Engineering

A Multi-Model View of Process Modelling

C. Rolland, N. Prakash and A. Benjamen
Université de Paris 1 Sorbonne, Centre de Recherche en Informatique, Paris, France

Situatedness of development processes is a key issue in both the software engineering and the method engineering communities, as there is a strong felt need for process prescriptions to be adapted to the situation at hand. The assumption of the process modelling approach presented in this paper is that process prescriptions should be selected according to the actual situation at hand, i.e. dynamically in the course of the process. The paper focuses on a multi-model view of process modelling which supports this dynamicity. The approach builds on the notion of a labelled graph of intentions and strategies called a map *as well as its associated* guidelines. *The map is a navigational structure which supports the dynamic selection of the intention to be achieved next and the appropriate strategy to achieve it, whereas guidelines help in the operationalisation of the selected intention. The paper presents the map and guidelines and exemplifies the approach using the CREWS-L'Ecritoire method for requirements engineering.*

Keywords: Method engineering; Process enactment mechanism; Process guidance; System development process modelling

1. Introduction

Process engineering is considered today as a key issue by both the software engineering and information systems engineering communities. Recent interest in process engineering is part of the shift of focus from the product to the process view of systems development. The belief of the software engineering community is that as a result of improved development processes [1–3] there will be both improved productivity of the software systems industry and improved systems quality. The focus has been to increase the level of formality of process models in order to make possible their enactment in process-centred software environments [4]. As a consequence, a large number of process models have been developed that Dowson [1] classifies as *activity-oriented* models, *product-oriented* models and *decision-oriented* models.

The software process modelling community realised quite early that even though process models were prescriptive, in actual practice departures from the prescription occurred [5–9]. Therefore, a concerted effort was put in to allow process models to respond to these departures. One approach was to assume prescriptive models and then modify them to accommodate real processes. This modification could be achieved in two ways. First the extent of deviations from the prescription that could be allowed was modelled as constraints [10–12]. Any actual deviation that satisfied the constraint was therefore manageable and the process enactment mechanism could handle it. This way of handling deviations took the prescriptive approach to its logical conclusion: it prescribed the deviations allowed in a prescription. The second way of handling deviations is to allow changes to be made in the prescription as and when they are needed [4,13–17]. Thus, a dynamic change of the basic prescription is allowed.

In recent years, the information systems community has concentrated on the need for adapting and extending existing methods to meet the changing needs of practice. Method engineering [18,19] represents the effort to improve the usefulness of systems development methods by creating an adaptation framework whereby methods are created to match specific organisational situations. This improvement has been attempted at two levels. At a global level, it deals with determining the project contingency factors [20,21] that help in selecting the

Correspondence and offprint requests to: C. Rolland, Université de Paris 1 Sorbonne, Centre de Recherche en Informatique, 17 rue de la Sorbonne, 75231 Paris Cedex 05, France. Email: rolland@univ-paris1.fr

right method to be used, whereas at a more fine-grained level it deals with on-the-fly construction of the process prescription fitting the situation at hand.

The latter was carried out in the contextual model [22–25]. Here the attempt was to relax the prescription given by a process model. Thus, the process model did not always specify what *must* be done but contained some specification of what *can* be done. The process model therefore contained a number of alternative ways of doing a task and a selection of the particular alternative was done dynamically, depending upon the situation in which the product was found. However, the contextual model could consist of both alternatives as well as prescriptions. Whenever such alternatives were available, the net effect was that the process model could be dynamically built, even as the process was being performed. The major difference between the software engineering approaches and the contextual approach is that whereas handling departures from prescriptions is an exception handling activity in the former, selection from alternatives in the latter is the normal activity envisaged in the process model itself and supported by a dynamic selection mechanism. Thus, support for real processes is provided in a more natural way.

In this paper, we propose to relax the prescription of a process model even further. Our proposal is based on the experience with the contextual model that we gained working with four groups of postgraduate students. The experiment consists of using the six methods described with the contextual model in Plihon [26] to develop application case studies within the process-centred environment MENTOR [14]. Our experience was that a key discriminant factor in real processes is the product situation. This situation has a strong bearing in selecting the task best suited to handle it and also the strategy to be adopted in carrying out this task. For example, consider a process for doing requirements engineering using goal–scenario coupling. Assume that a goal G has been elicited. Now, it is possible either to explore alternative goals of G or to write a scenario for it. Thus, the process model must reflect this choice and the requirements engineer would dynamically choose between one of these alternatives. It can be seen that G provides a basis for a discriminant choice in what task is to be done next. Now, consider that a fully developed scenario has been written out and goals are to be determined by scenario analysis. That is, the next task to be done is known. However, it is possible to discover goals that are exceptions or obstacles to G or sub-goals of G using the alternative or the composition discovery strategies. Again, these strategies for eliciting goals need to be reflected in the process model so that the right one can be dynamically chosen depending on the nature of the scenario. Thus, the product situation also provides a basis for a discriminant choice in what strategy is to be adopted in performing a task. Evidently, a process model that captures all alternatives of tasks and strategies is needed to support processes. Such a model needs to be backed up by a dynamic selection mechanism of tasks and strategies. In the paper we propose to represent task and strategies alternatives as a labelled directed graph called a *map* and provide support in alternative selection through guidelines.

It can be seen that the salient features of our approach are:

1. explicit recognition of the role of strategies in process modelling;
2. a non-prescriptive model of strategies and tasks containing alternatives only from which real processes can be built;
3. dynamic process construction is the rule rather than an exception.

As indicated above, the non-prescriptive model is a labelled directed graph called a *map*. The map uses two fundamental notions: a process intention or *intention* for brevity, and *strategy*. An intention captures in it the notion of a task that the application engineer intends to perform whereas the strategy is the manner in which the intention can be achieved. The *nodes* of the map are intentions whereas the *edges* are labelled with strategies. The directed nature of the map identifies which intention can be done after a given one. The only way in which a process can be built is dynamically, through the use of guidelines for selection among alternatives. Only after the task and the strategy have been decided is there a need for a guideline to achieve the task.

There are three guidelines associated with the map:

- intention selection guidelines for determining all succeeding intentions of a given one;
- strategy selection guidelines for determining the strategies from which one is selected;
- intention achievement guidelines for defining the way in which an intention can be achieved. Thereafter, the enactment mechanism is invoked to actually carry out the tasks.

We view a map as containing a panel of process prescriptions from which, by dynamic selection, the particular one that is best suited to the product situations as they emerge is selected. In this sense, the map is a *multi-model* with dynamic process modelling capability.

The layout of the paper is as follows. In the next section the notion of the map as a labelled directed graph is presented and the multi-model capability of the map is highlighted. In Section 3, the different kinds of guidelines and their structure are considered. The manner in which guidelines relate to the map is articulated.

Section 4 contains the representation of the CREWS-L'Ecritoire method as a map of guidelines. This serves as an example to illustrate how the map and guidelines can be used to represent real methods. Section 5 deals with the meta-process, i.e. the process to develop and enact application processes. The use of the meta-process to develop the requirements specification of a recycling machine is presented in Section 6. Section 7 is the concluding section.

2. The Map

A map is a *process model* which is associated with a *product model* as shown in Fig. 1 to form a method. Figure 1 describes our method view using an E/R-like notation. A box represents an entity type (ET), the labelled link represents a relationship type (RT) and the embedded box refers to an objectified RT. Multiplicities are denoted with couples of minimum and maximum cardinality values.

A map is a process model in which a non-deterministic ordering of intentions and strategies has been included. It is a labelled directed graph with intentions as nodes and strategies as edges between intentions. The directed nature of the graph shows which intentions can follow which one. Figure 2 describes the map meta-model using the same E/R-like notation as above. As shown in the figure, a map consists of a number of *sections* each of which is a triplet <I_i,I_j,S_{ij}>.[1] There are two distinct intentions called *Start* and *Stop* respectively that represent the intentions to start navigating in the map and to stop doing so. Thus, it can be seen that there are a number of paths in the graph from *Start* to *Stop*.

We assume development processes to be intention-oriented. At any moment, the application engineer has an *intention*, a goal in mind that he/she wants to fulfil. To take this characteristic into account the map identifies the set of intentions that have to be achieved in order to solve the problem at hand.

Let I be this set.

An intention is a goal, an objective that the application engineer has in mind at a given point of time. An intention statement expressed in natural language usually starts with a *verb* and may comprise several *parameters*, where each parameter plays a different role with respect to the verb. The key parameter is the *target* of the verb; for example, in the examples below, *Scenario* and *Goal* are the targets of the verbs *Conceptualise* and *Elicit* respectively.

[1]Intentions are in italics (I_i, I_j). Strategies are in 'Arial' font (S_{ij}).

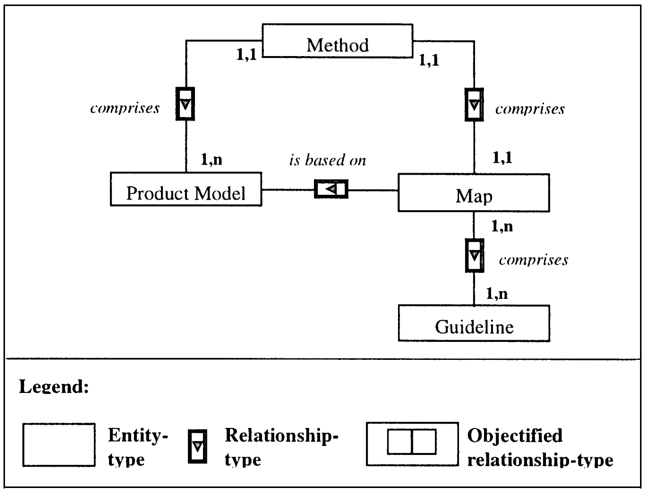


Fig. 1. Map and product model.

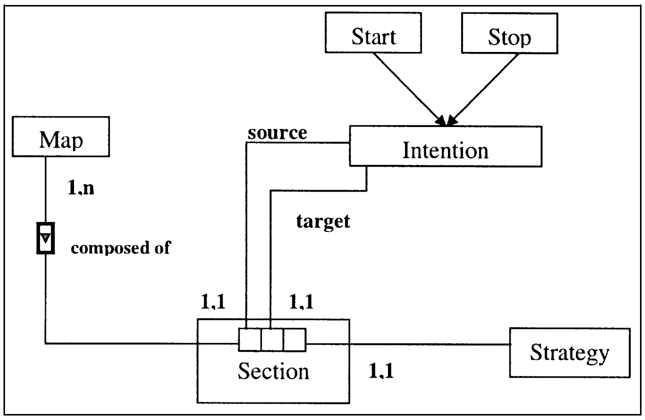


Fig. 2. The map meta-model.

(a) *Conceptualise*$_{\text{verb}}$ *a Scenario*$_{\text{object}}$
(b) *Elicit*$_{\text{verb}}$ *a Goal*$_{\text{result}}$

As shown in the examples above, there are two types of targets, *Objects* and *Results*. Both refer to product parts, i.e. elements of the product model, which are either objects or subjects of the process intention. An *Object* is supposed to exist before the goal is achieved. For example, in the goal statement (a) the target *Scenario* is an object because it exists even before *Conceptualise* is achieved. In contrast, a *Result* results of the achievement of the intention. For example, in the goal statement (b), a *Goal* is the result of the achievement of the intention *Elicit*. We shall introduce other parameters of the verb in an intention statement as needed in the paper. For more details see Prat [27] and Rolland et al. [28].

A *strategy* is an approach, a manner to achieve an intention. The strategy, as part of the triplet <I_i,I_j,S_{ij}>, characterises the flow from I_i to I_j and the way I_j can be achieved.

Let S be the set of strategies identified in the map.

It can be seen that the map can represent in it all the meaningful interconnections between process intentions and strategies. Formally, the map is a subset of the Cartesian product:

$$\mathbf{Map} \subseteq \mathbf{I} \times \mathrm{I} \times \mathbf{S}$$

The specific manner in which an intention can be achieved is captured in a section of the map whereas the various sections having the same intention I_i as a source and I_j as target show the different strategies that can be adopted for achieving I_j when coming from I_i. Similarly, there can be different sections having I_i as source and I_{j1}, I_{j2}, …, I_{jn} as targets. These show the different intentions that can be achieved after the achievement of I_i.

Let there be two map sections, MS1 and MS2. MS1 and MS2 are connected in the map provided the target intention of MS1 is the source intention of MS2. For example, the sections $<I_i,I_j,S_{ij}>$ and $<I_k,I_i,S_{ki}>$ are interconnected in the map because the target intention I_i of the latter is also the source intention of the former. Thus, I_j is reachable from I_k through the intermediate intention I_i.

As an example consider Fig. 3, which contains six sections MS0 to MS5 having connections at I_i, I_j and I_k.

As shown in the figure, there might be several flows from I_i to I_j, each corresponding to a specific strategy (for example, MS1 and MS2 in Fig. 3). In this sense the map offers *multi-thread flows*. There might also be several strategies from different intentions to reach an intention I_i (for example, MS3 and MS4 in Fig. 3). In this sense the map offers *multi-flow paths* to achieve an intention. Finally, the map can include reflexive flows (see MS3 in Fig. 3).

A map is a *navigational structure* in the sense that it allows the application engineer to determine a path from *Start* intention to *Stop* intention. The map contains a finite number of paths, each of them prescribing a way to develop the product, i.e. each of them is a process model. Therefore the map is a *multi-model*. It embodies several process models, providing a multi-model view for modelling a class of processes. None of the finite set of models included in the map is recommended 'a priori'. Instead the approach suggests a dynamic construction of the actual path by navigating in the map. In this sense the approach is sensitive to the specific situations as they arise in the process. The next intention and strategy to achieve it are selected dynamically by the application engineer among the several possible ones offered by the map. Furthermore, the approach is meant to allow the dynamic adjunction of a path in the map, i.e. adding a new strategy or a new section in the actual course of the process.

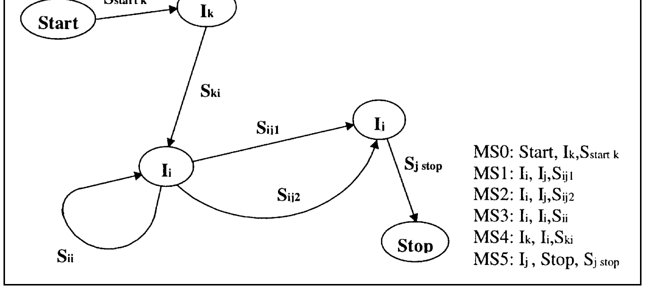

Start
$S_{start\ k}$
I_k
S_{ki}
S_{ij1}
I_j
I_i
$S_{j\ stop}$
S_{ij2}
S_{ii}
Stop
MS0: Start, I_k,$S_{start\ k}$
MS1: I_i, I_j,S_{ij1}
MS2: I_i, I_j,S_{ij2}
MS3: I_i, I_i,S_{ii}
MS4: I_k, I_i,S_{ki}
MS5: I_j , Stop, $S_{j\ stop}$

Fig. 3. Examples of map sections.

In such a case guidelines that make available all choices open to handle a given situation are of great convenience. The map is associated to such guidelines. These are presented in the next section.

3. Guidelines

A guideline is defined [29] as 'a set of indications on how to proceed to achieve an objective or perform an activity'. For us, a guideline embodies *method knowledge* to guide the application engineer in achieving an intention in a given situation. In this section we first consider the different kinds of guidelines and their relationships to the map. Thereafter the structure of the guidelines as comprising a *signature* and a *body* is considered and the relationship between the guideline signature and the kind of guideline is brought out.

3.1. Kinds of Guidelines

As shown in Fig. 4, we associate the map with guidelines, namely one '*Intention Achievement Guideline*' per section $<I_i,I_j,S_{ij}>$, one '*Intention Selection Guideline*' per node I_i, except for *Stop*, and one '*Strategy Selection Guideline*' per node pair $<I_i,I_j>$.We will refer to them as IAG, ISG and SSG respectively.

An intention-driven process is an iterative process that repeatedly resolves two issues, namely, (1) how to fulfil the intention he/she reached and (2) how to select the right section to progress. IAGs support the former whereas ISGs and SSGs help in the latter. More precisely:

1. There exists an *Intention Achievement Guideline* (*IAG*) for every triplet $<I_i,I_j,S_{ij}>$. It aims at supporting the application engineer in the achievement of intention I_j according to the strategy S_{ij}.

For a section $<I_i,I_j,S_{ij}>$, there is an IAG.

An IAG provides an operational means to fulfil the intention. This means that an IAG implies the transformation of the product under development. Whereas the map identifies strategies to reach intentions,

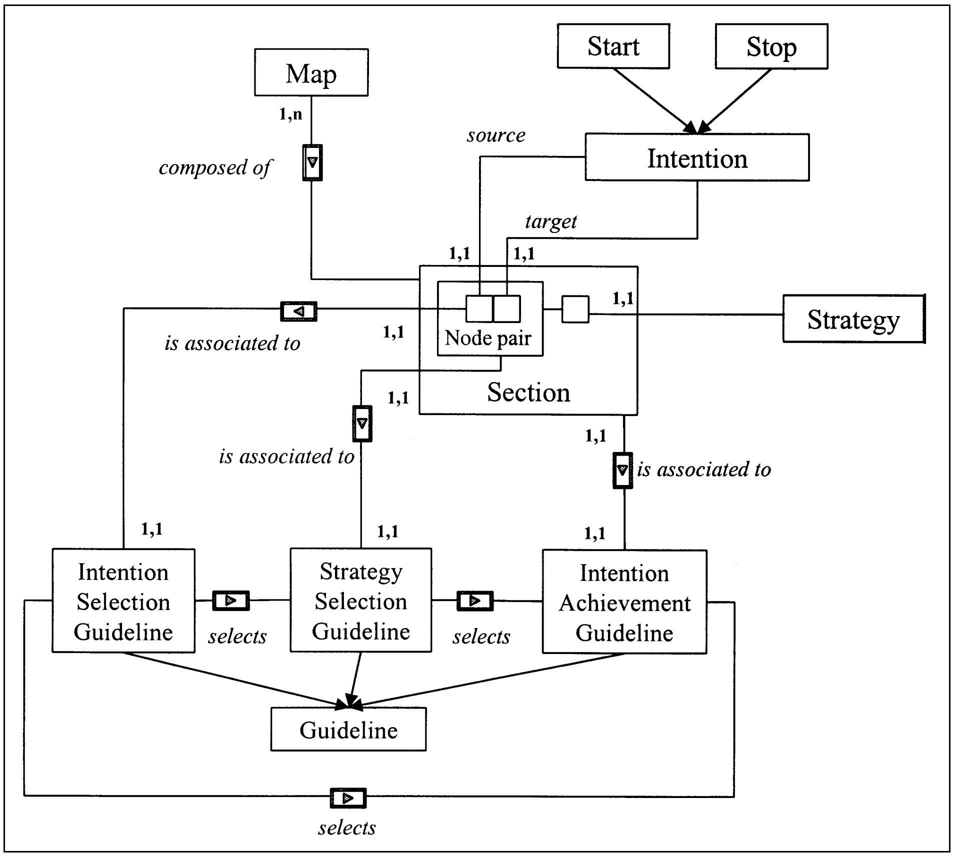


Fig. 4. The map guideline relationships.

IAGs are concerned with the *tactics* to implement these strategies. There might be several tactics offered by an IAG. This means that an IAG may contain alternative operational ways to fulfil the intention. Besides, it might be necessary to proceed in a number of steps to reach the ultimate effect of an IAG, that is, to perform some action on the product under development. Consequently an IAG may include the decomposition of the initial intention into sub-intentions which themselves may be decomposed until intentions executable through actions on the product are reached. Therefore, an IAG may be seen as a goal tree which helps in performing the operationalisation of an intention I through sub-intentions connected by alternative and decomposition relationships into actions on the product.

2. Given two Intentions I_i, I_j and a set of possible strategies S_{ij1}, S_{ij2}, …, S_{ijn} applicable to I_j, the role of the *Strategy Selection Guideline* (SSG) is to guide the selection of an S_{ijk}, thereby leading to the selection of the corresponding IAG.

For a node pair <I_i,I_j>, there is an SSG.

An SSG first determines all the strategies that can be used to achieve I_j from I_i. It does this by the operation SOP, *Strategy Operator*, defined as follows:

SOP : I × I→ {S | <I,I,S> is a section}

For example, in the map of Fig. 3:

SOP (I_i,I_j) ={S_{ij1},S_{ij2}}

The set of strategies is presented by SSG to the application engineer, who picks the one most appropriate to the situation at hand. Thus, the section <I_i,I_j,S_{ijk}>is selected. Since a unique Intention Achievement Guideline is associated with each section, the SSG determines this. The enactment mechanism then performs I_j according to the selected strategy in the task organisation specified by the Intention Achievement Guideline.

3. Given an intention I_i, an *Intention Selection Guideline* (ISG) identifies the set of intentions {I_j} that can be achieved in the next step and selects the corresponding set of either IAGs or SSGs. The former is valid when there is only one section between I_i and I_j, whereas the latter occurs when there are several sections between I_i and I_j.

For an intention I_i, there is an ISG.

An ISG first determines all the intentions that can be done after a given one. It does this through the operation IOP, *Intention Operator*, defined as follows:

IOP : I → {I | <I,I,S> is a section}

That is, IOP determines the set of intentions which are the target intentions of sections having the same source intention.

For example, in the map of Fig. 3:

IOP (I_i) ={I_j, I_i}

The application engineer then picks one intention from these – that which is most appropriate for the situation at hand. The ISG then determines whether there is only one section between the source and the selected target intention or whether there are several sections. In the former case, the IAG associated with the section is used by the enactment mechanism to achieve the target intention. In the case where several sections exist between the source and the selected target intention, the SSG is invoked to determine the strategy to be used in the situation which, as discussed earlier, leads to the determination of an IAG and subsequent enactment. In our example, IOP has determined two target intentions I_j and I_i as shown above. There is only one section between the source intention I_i and the target I_i. This is <I_i,I_i,S_{ii}>. Thus, if the application engineer chooses I_i as the target, then the IAG is determined. ISG can cause intention achievement with no further intervention from the application engineer. On the other hand, there are two sections having I_i as source and I_j as target. These are <I_i,I_j,S_{ij1}> and <I_i,I_j,S_{ij2}> respectively. If the application engineer chooses I_j as the target intention then SSG must be used to decide which of these will be used. The IAG is determined and I_j achieved.

It can be seen from the foregoing that the objective of the ISGs is met by placing reliance upon SSGs and IAGs. Similarly SSGs rely on IAGs. Therefore, determination of the intention to handle a given situation, determination of the strategy to be adopted and the task organisation are all integrated together.

Summarising, then, Fig. 5 associates the ISGs, IAGs and SSGs with the map shown in Fig. 3. There are six IAGs, one per section, four ISGs for each of the nodes except *Stop*, and four SSGs for each of the four node pairs <I_i, I_j>.

3.2. Structure of a Guideline

Even though there are different kinds of guidelines, all of these depict the same underlying structure. Figure 6 shows the guideline meta-model expressed again in an E/R-like notation. Our proposal for the description of a guideline relies on the NATURE contextual approach [23,30] and its corresponding enactment mechanism [14,31]. As shown in Fig. 6, a guideline has a *body* which encapsulates method knowledge and a *signature*. We consider these in turn.

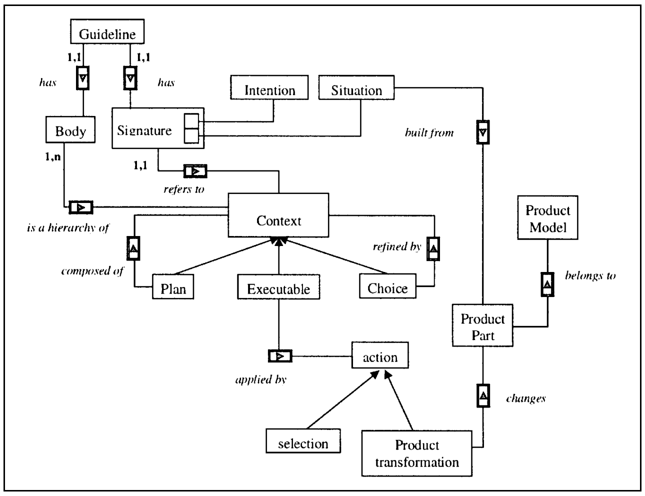


Fig. 6. The guideline meta-model.

Map section	IAG Reference
MS0: *Start*, I_k,$S_{start\ k}$	IAG0
MS1: I_i, I_j,S_{ij1}	IAG1
MS2: I_i, I_j,S_{ij2}	IAG2
MS3: I_i, I_i,S_{ii}	IAG3
MS4: I_k, I_i,S_{ki}	IAG4
MS5: I_j , *Stop*, $S_{j\ stop}$	IAG5

Intention	ISG Reference
Start	ISG0
I_i	ISG1
I_j	ISG2
I_k	ISG3

Node pair	SSG Reference
Start, I_k	SSG0
I_k, I_i	SSG1
I_i, I_j	SSG2
I_j, *Stop*	SSG3

Fig. 5. Guidelines of the map presented in Fig. 3.

3.2.1. Guideline Signature

A signature is a pair <(sit), *I*>, where (sit) is the situation and *I* is an intention. For example, <(Goal), *Author Scenario*> is a signature. The situation refers to the product under development and the intention is the goal that the application engineer wants to achieve in this situation. In the previous example the situation is the product part 'Goal' and *Author Scenario* is the intention *I* that the application engineer wants to achieve. The three kinds of guidelines, namely ISGs, SSGs and IAGs, have signatures of the generic form <(sit), *I*>. However (sit) and *I* can be specialised for each of the three kinds of guidelines. This is summed up in Fig. 7 and explained below.

First, as mentioned earlier, the map identifies two issues to be solved by the application engineer: (a) how to perform the intention he/she has reached, and (b) how to select the right section to progress further. This leads to an identification of two major classes of intentions of signatures: the *Achieve* and the *Progress*. As IAGs support issue (a), the signature intention of a IAG refers to a process achievement intention and therefore belongs to the *Achieve* signature intention class. SSGs and ISGs which help in (b) have signature intentions which express process progression towards process achievement and, therefore, belong to the *Progress* signature intention class. Therefore, we propose to use the map intention *I* in IAG intention signatures and the generic term *Progress* as intention signature for SSGs and ISGs.

Second, we propose to differentiate an SSG intention signature from an ISG one using the statement $\textit{Progress}_{\text{verb}}$ $(\textit{from } I_i)_{source}$ for the former and $\textit{Progress}_{\text{verb}}$ $(\textit{to } I_j)_{target}$ for the latter.

$\textit{Progress}_{\text{verb}}$ $(\textit{from Author Scenario})_{\text{source}}$

and

$\textit{Progress}_{\text{verb}}$ $(\textit{to Author Scenario})_{\text{target}}$

are two examples of signature intentions belonging to the class *Progress*. As shown in these examples, *Progress* is the verb of the intention statement, (from *Author Scenario*) is the *source parameter* of the verb and (to *Author Scenario*) corresponds to the *target parameter*.

Third, we suggest integrating the name of the strategy in the statement of the achievement intention of an IAG. Therefore, the IAG for a section $<I_i, I_j, S_{ij1}>$ has an intention signature of the form I_j *with* S_{ij}.

$\textit{Author}_{verb}$ $\textit{Scenario}_{result}$ $(\textit{with linguistic strategy})_{manner}$

is an example of intention belonging to the class *Achieve*. As indicated in the intention statement *Author* is the verb, *Scenario* is its *result* and (*with linguistic strategy*) corresponds to the *parameter manner*.

Finally, the situation part of the guideline signature refers to the product part(s) resulting from the achievement of the start intention (I_i) of the map section associated to the guideline. We will see in the next section that the situation may include constraints on the product. These constraints on (sit) play the role of a precondition for the intention *I* to be achievable. It can be seen that the guideline establishes the connection between the process and the product models, making precise the part of the product (and its associated constraints) influencing the process flow.

(Scenario)

and

(Scenario: state (Scenario) = written)

are two examples of situations. In the first case (Scenario) refers to the product part 'Scenario', whereas in the second case the situation constrains the 'Scenario' to be in the state 'written'.

3.2.2. Guideline Body

The body describes the way in which *Achieve* and *Progress* intentions are fulfilled. Following the contextual approach the body is organised around the notion of a context that can be of three different types: *executable*, *plan*, *choice*; and two types of relationships

Type of guideline	Map reference	Guideline signature
IAG_i	$< I_i, I_j, S_{ij}>$	(sit*(I_i), I_j *with* S_{ij})
ISG_i	$< I_i >$	(sit (I_i), *Progress from* I_i)
SSG_i	$< I_i, I_j >$	(sit (I_i), *Progress to* I_j)

*Sit(I_i) refers to the product situation after I_i has been achieved.
Progress refers to a class of intentions in order to progress in the process.
In contrast I_j, I_i are achievement intentions.

Fig. 7. Correspondence between the kind of guideline and the guideline signature.

among contexts: *composition* and *refinement* (Fig. 6). The latter leads to an organisation of a guideline as a hierarchy of contexts connected by AND (composed of) and OR (refined by) relationships. The former helps in distinguishing situations offering choices (choice contexts) from those which require decomposition of contexts (plan contexts). Executable contexts are of two types: in IAGs they are associated to actions which transform the product under development. The guideline is therefore a means to articulate the consequences of satisfying the intention of the guideline signature on the product under development. In SSGs and ISGs they perform actions to select IAGs. The enactment mechanism takes care of the presentation of available choices, the performance of plan contexts and of the impact of the execution of actions on the product under construction. For further details on the contextual approach see elsewhere [23,32–35].

4. A Multi-Model View of CREWS-L'Ecritoire

This section instantiates the map meta-model presented in Section 2 with the goal—scenario method for requirements engineering developed in the CREWS project [28,36–38].

4.1. Overview of the CREWS-L'Ecritoire Approach

The method combines a *goal-driven approach* to requirements engineering with the *use of scenarios*. The total solution is in two parts. First, for a goal, scenarios are authored by the scenario author. Thereafter, the authored scenario is explored to yield goals which, in turn, cause new scenarios to be authored, and so on.

As illustrated in Fig. 8, the RE process consists of repeating a two-phase cycle composed of (1) scenario authoring and (2) goal discovery. The resulting product is a hierarchy of pairs (G, Sc), where G is a goal and Sc a scenario. Each pair is called a requirements chunk (RC). RCs are related to one another in three different ways through composition, alternative and refinement relationships. The composition and alternative relationships lead to an AND/OR structure between RCs, whereas the refinement relationship is used to describe RCs at different levels of abstraction (Fig. 8). A brief overview of the concepts and terminology of the CREWS product model is as follows:

- A ***requirement chunk*** (RC) is a pair <G, Sc>, where G is a goal and Sc is a scenario. Since a goal is intentional and a scenario is operational in nature, a requirement chunk is a possible way of achieving the goal.

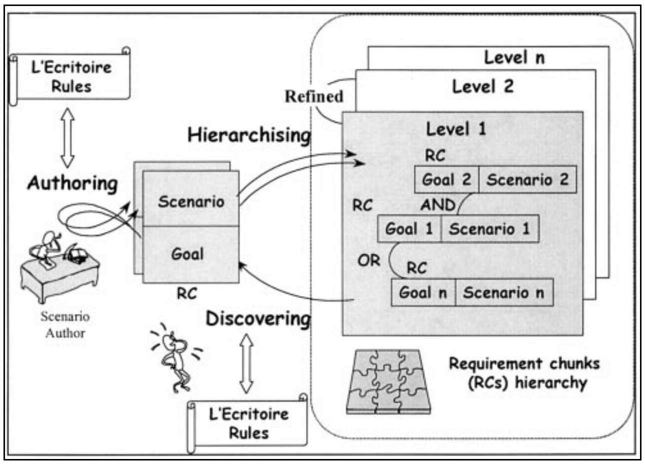


Fig. 8. Overview of the CREWS RE process.

- A ***goal*** is defined as 'something that some stakeholder hopes to achieve in the future'. In our approach, a goal (similar to an intention map) is expressed as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb. An example of a goal expressed in this structure is the following:

Provide$_{verb}$ *(efficiently)*$_{quality}$ *(electricity)*$_{target}$ *(from EDF producer)*$_{source}$ *(to our non-eligible customers)*$_{beneficiary}$ *(using the EDF network)*$_{means}$

- A ***scenario*** is 'a possible behaviour limited to a set of purposeful interactions taking place among several agents'. It is composed of one or more *actions*, an *action* being an interaction *from* one agent *to* another. The combination of actions in a scenario describes a unique path. A scenario is characterised by initial and final states. An *initial state* attached to a scenario defines a precondition for the scenario to be triggered. A *final state* defines a state reached at the end of the scenario. We distinguish between *normal* and *exceptional* scenarios. The former leads to the achievement of its associated goal whereas the latter fails in goal achievement.
- ***Relationships between requirement chunks***: there are three types of relationships among requirement chunks, namely, the composition, alternative, and refinement relationships. The first two of these lead to a horizontal AND/OR structure between RCs. These are extensions of conventional AND/OR relationships between goals. AND relationships among RCs link together those chunks that require each other to define a completely functioning system. RCs related through OR relationships represent alternative ways of ful-

filling the same goal. The third kind of relationship relates requirement chunks at different levels of abstraction. The refinement relationship establishes a vertical link between requirement chunks.

4.2. The CREWS-L'Ecritoire Map and Guidelines

As shown in Fig. 8, the RE process is supported by automated rules embodied in a computer-based software tool called *L'Ecritoire*. Automated rules act in the two phases of the goal-discovery, scenario-authoring, goal-discovery cycle to respectively guide scenario authoring and help in discovering goals.

The corresponding map and guidelines are presented in Fig. 9(a) and (b) respectively.

As can be seen, the map of Fig. 9(a) provides a number of paths for going from *Start* to *Stop*. The sequence '*Start*, linguistic strategy to *Elicit a Goal*, free prose to *Write a Scenario*, manual strategy to *Conceptualise a Scenario,* completeness strategy to *Stop*' is a path. Another path could be the one which after *Conceptualise a Scenario* uses the composition discovery strategy to achieve *Elicit a Goal* and then goes to *Stop* through case-based discovery to *Elicit a Goal*, free prose to *Write a Scenario*, manual strategy to *Conceptualise a Scenario*, completeness strategy to *Stop*. It is evident that each of these paths is a process model. The multiple process models that can be generated from the map are limited only by the map itself.

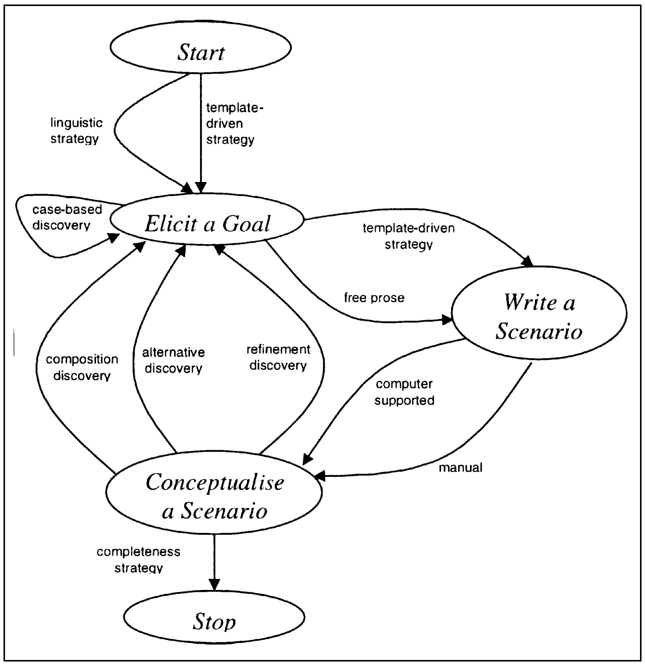


Fig. 9(a). Map of the CREWS-L'Ecritoire method.

The generation of an actual process model is not done in any ad hoc way but is driven by the situation of the product after an intervention has been achieved. For example, after achievement of *Elicit a Goal*, the situation could be that case-based discovery strategy is used to again *Elicit a Goal*. The resulting situation, after *Elicit a Goal*, could now ask for the free prose strategy to be used to *Write a Scenario*. The point is that the process model is shaped dynamically by the situations which arise as a result of intention achievement. This means that the time gap between process model generation and process enactment is reduced to zero. This facilitates changes in the process model as the process is performed.

Process model generation is under the control of guidelines. For instance, SSG4 supports the selection of the linguistic strategy *to Elicit a Goal* in the first path presented above. ISG1 thereafter helps in the selection of *Write a Scenario,* whereas SSG3 supports the selection of the free prose strategy for achieving it. The section (*Elicit a Goal*, *Write a Scenario*, free prose) is now selected and IAG8 supports the achievement of *Write a Scenario*. The use of guidelines continues until the entire process model has been generated.

There is an intention achievement guideline for each of the 11 sections of the map of Fig. 9(a). Five SSGs are associated with the five node pairs *Elicit a Goal–Write a Scenario*, *Write a Scenario–Conceptualise a Scenario, Conceptualise a Scenario–Elicit a Goal*, *Start–Elicit a Goal* and *Conceptualise a Scenario–Stop*. Additionally, there are four ISGs, one for each of the map intentions: *Start*, *Stop*, *Elicit a Goal* and *Conceptualise a Scenario*. Figures 10, 11 and 12 give three examples of guidelines, one for each type.

4.2.1. IAG8 Example

As an intention achievement guideline, IAG8 provides advice to the requirements engineer to achieve the goal *Write a Scenario in free prose*.

The guideline is characterised by its signature, <(sit),I>, which expresses the intention to be fulfilled (*Write a Scenario in free prose*) and the situation required for the intention-to-be-fulfilled goal (G).

The situation refers to the goal part of the product under development (i.e., the RCs hierarchy) whereas the intention is a sub-type of the *Achieve* signature intention of Section 3. The body is a two-level hierarchy of contexts (Fig. 10). The first level is a plan context suggesting two steps to write a scenario:

1. to get writing guidance if desired;
2. to write the scenario itself.

Intention Achievement Guidelines (IAG)

<(G), *Elicit a Goal with case-based discovery strategy*>	**IAG1**
<(RC: state (RC) = completed), *Elicit a Goal with composition strategy*>	**IAG2**
<(RC: state (RC) = completed), *Elicit a Goal with alternative strategy* >	**IAG3**
<(RC: state (RC) = completed), *Elicit a Goal with refinement strategy* >	**IAG4**
<(Stat.), *Elicit a Goal with linguistic strategy* >	**IAG5**
<(Stat.), *Elicit a Goal with template-driven strategy*>	**IAG6**
<(G), *Write a Scenario with template-driven strategy* >	**IAG7**
<(G), *Write a Scenario in free prose*>	**IAG8**
<(Sc: state (Sc) = written), *Conceptualise a Scenario with computer support strategy*>	**IAG9**
<(Sc), *Conceptualise a Scenario manually*>	**IAG10**
<(RCs: state (RCs) = completed), *Stop with completeness strategy*>	**IAG11**

Strategy Selection Guideline

<(RC: state (RC) = completed), *Progress to Elicit a Goal*>	**SSG1**
<(Sc: state (Sc) = written), *Progress to Conceptualise a Scenario*>	**SSG2**
<(G), *Progress to Write a Scenario*>	**SSG3**
<(Stat.), *Progress to Elicit a Goal*>	**SSG4**
<(RCs: state (RCs) = completed), *Progress to Stop*>	**SSG5**

Intention Selection Guideline

<(G), *Progress from Elicit a Goal*>	**ISG1**
<(RC: state (Sc) = completed), *Progress from Conceptualise a Scenario*>	**ISG2**
<(Sc: state (Sc) = written), *Progress from write a Scenario*>	**ISG3**
<(Stat.), *Progress from Start*>	**ISG4**

Fig. 9(b). Guidelines of the CREWS-L'Ecritoire method.

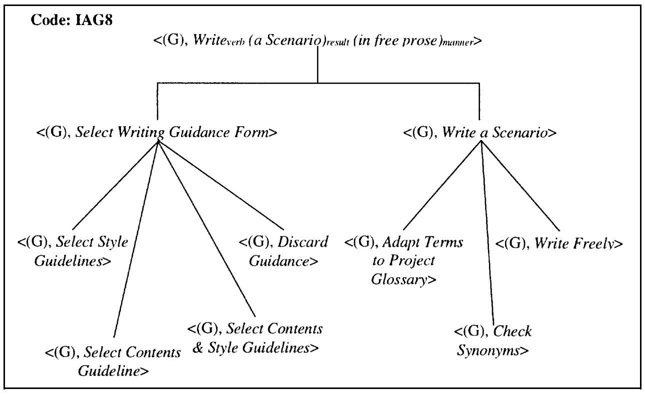

Code: IAG8
<(G), Write$_{verb}$ (a Scenario)$_{result}$ (in free prose)$_{manner}$>
<(G), Select Writing Guidance Form>
<(G), Write a Scenario>
<(G), Select Style Guidelines>
<(G), Discard Guidance>
<(G), Adapt Terms to Project Glossary>
<(G), Write Freely>
<(G), Select Contents & Style Guidelines>
<(G), Select Contents Guideline>
<(G), Check Synonyms>

Fig. 10. Example of Intention Achievement Guideline.

Each of these steps is a component context of the plan, namely, <(G), *Select Writing Guidance Form*> and <(G), *Write a Scenario*>, which both offer choices.

Indeed, in the CREWS-L'Ecritoire approach, the requirements engineer has the possibility to use style guidelines, contents guidelines, both of them or to discard any proposed guidance. Style guidelines recommend a style of writing whereas contents guidelines define the semantics of the scenario contents. These choices are expressed in the choice context <(G), *Select Writing Guidance Form*>.

The choice context <(G), *Write a Scenario*> offers three options:

(a) alignment of the terms used in the scenario with a general project glossary;
(b) detection and possible removal of synonyms,
(c) without any control.

All the leaves of the hierarchy are executable contexts.

4.2.2. SSG1 Example

A Strategy Selection Guideline such as SSG1 has a signature <(sit), *I*> which expresses that the requirements engineer wants to progress in the RE process by achieving intention *I* in a given situation (sit). The intention is a sub-type of the *Progress* signature intention of Section 3. The SSG1 signature, <(RC: State (RC) = completed), *Progress to Elicit a Goal*>, associates the intention of progressing towards the target to *Elicit a Goal* when the requirement chunk (RC) has been completed. Notice that in this case the situation associates a constraint to the product part (Requirement Chunk) it refers to. The body of SSG1 is a hierarchy of

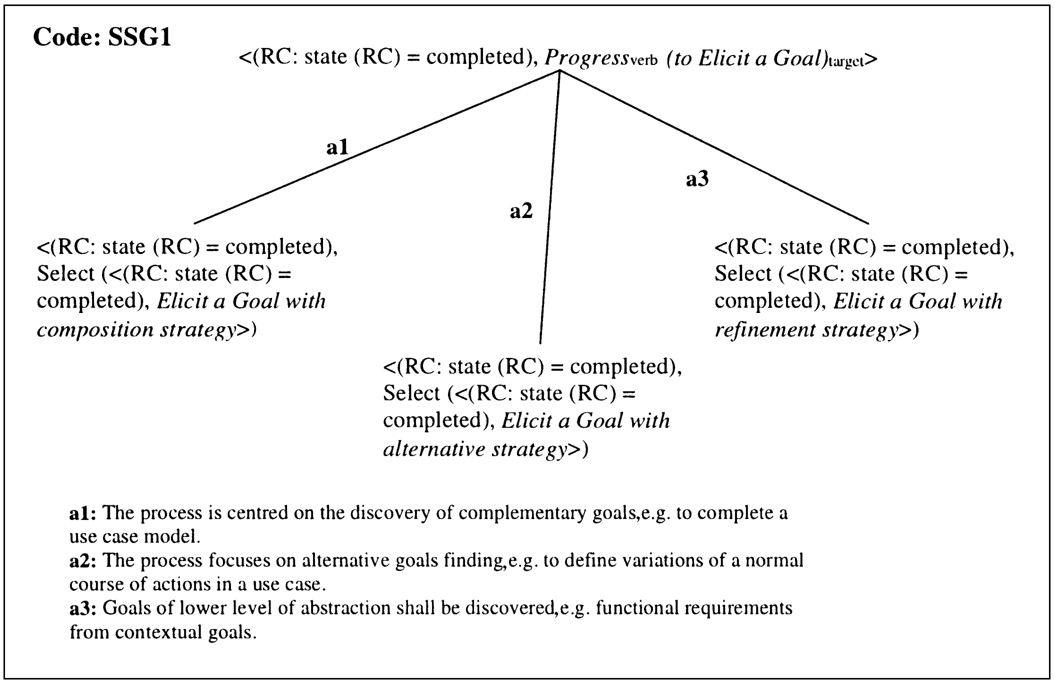

Code: SSG1
<(RC: state (RC) = completed), *Progress*verb *(to Elicit a Goal)*target>
a1
a2
a3
<(RC: state (RC) = completed), Select (<(RC: state (RC) = completed), *Elicit a Goal with composition strategy*>)
<(RC: state (RC) = completed), Select (<(RC: state (RC) = completed), *Elicit a Goal with alternative strategy*>)
<(RC: state (RC) = completed), Select (<(RC: state (RC) = completed), *Elicit a Goal with refinement strategy*>)
a1: The process is centred on the discovery of complementary goals,e.g. to complete a use case model.
a2: The process focuses on alternative goals finding,e.g. to define variations of a normal course of actions in a use case.
a3: Goals of lower level of abstraction shall be discovered,e.g. functional requirements from contextual goals.

Fig. 11. Example of Strategy Selection Guideline.

contexts having the signature of SSG1 as its root. SSG1 is a choice context offering three alternatives (Fig. 11). Each of these proposes the selection of an *Intention Achievement Guideline* to discover goals respectively following the composition strategy (*Select* <(RC: state (RC) = completed), *Elicit a Goal with composition discovery strategy l*>) or the refinement strategy (*Select* <(RC: state (RC) = completed), *Elicit a Goal with refinement discovery strategy*>) or the alternative strategy (*Select* <(RC: state (RC) = completed), *Elicit a Goal with alternative discovery strategy*>). Arguments (a1, a2, a3) are proposed to guide the requirements engineer in the selection of the appropriate strategy and associated guideline.

4.2.3. ISG1 Example

An *Intention Selection Guideline* is similar to a Strategy Selection Guideline in the sense that it guides the application engineer in progressing in the process. Thus, its signature contains an intention of the *Progress* type for a given situation (sit) which refers to a product part. The difference lies in the nature of the *Progress* intention which refers here to a 'source' intention, whereas it was a 'target' intention in the case of an SSG. For example, in ISG1 the intention is to progress from the source intention *Elicit a Goal*, i.e. when a goal has been elicited without any specific target intention in mind.

The body of an ISG offers all the possibilities to progress from the source intention and guides in the selection of either SSGs or IAGs as described in Section 3. For example, the ISG1 body (Fig. 12) is a choice context which offers two alternatives: the first one suggests to proceed with the case-based discovery strategy and proposes the selection of IAG1 (<(G), *Discover a Goal with case-based discovery strategy*>). The second one suggests a choice among the two strategies to *Write a Scenario* and proposes the selection of the SSG3 <(G), *Progress to Write a Scenario*>. Arguments a4 and a5 help in the choice of the more appropriate option for a given situation.

4.3. Experiments of the Approach

Besides being applied in the CREWS-L'Ecritoire approach to requirements engineering, the multi-model view presented here has served as a basis for representing (a) the three other requirements engineering approaches developed within the CREWS project, namely, the Real World Scenes approach [38], the SAVRE approach for scenario exceptions discovery [39] and the scenario animation approach [40] and (b) for integrating approaches [41] one with the other and with the OOSE approach [15]. In totality this has resulted in 18 maps and almost 100 guidelines. A report on these is

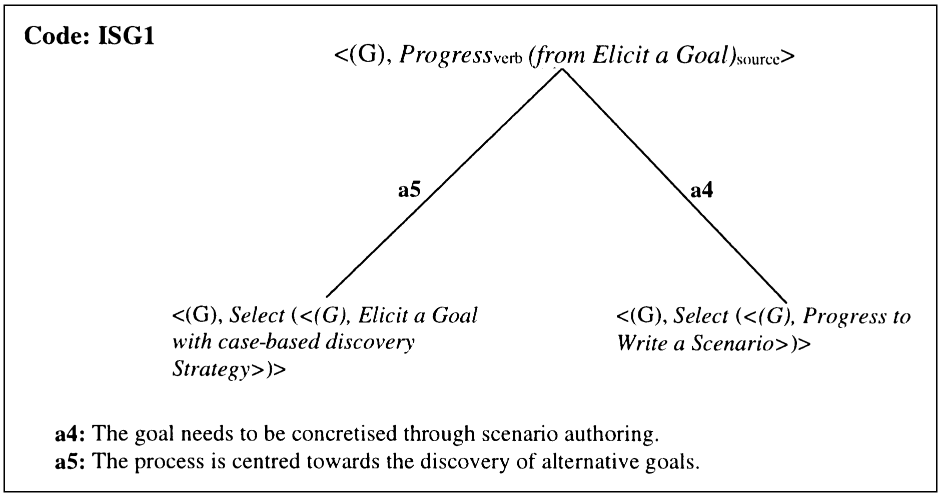

Code: ISG1
<(G), Progressverb (from Elicit a Goal)source>
a5
a4
<(G), Select (<(G), Elicit a Goal with case-based discovery Strategy>)>
<(G), Select (<(G), Progress to Write a Scenario>)>
a4: The goal needs to be concretised through scenario authoring.
a5: The process is centred towards the discovery of alternative goals.

Fig. 12. Example of Intention Selection Guideline.

under preparation and is expected to be available in the electronic CREWS method base [42] from September 1999.

As another important case study of the validation of the multi-model view of process modelling presented here, we would like to mention the electronic guide book to support the EKD-CM method, which is a specialisation of the Enterprise Knowledge Development method to managing Change Management in organisations [43].

Let us now turn our attention towards the process for enacting map and guidelines i.e. the meta-process.

5. The Meta-Process

As in Rolland [44], we define a *meta-process* as a process for the construction of a process model. In our case, the meta-process is a process for the generation of a path from the map and its instantaneous enactment for the application at hand. A meta-process is an instantiation of a model, the *meta-process model*. The meta-process model can be represented in many different ways and we choose here the map as a means to do so. In order to avoid ambiguity we shall refer to the map of the meta-model as the *meta-map* and to the map of the method as the *method map*.

As shown in Fig. 13, the meta-map consists of the four meta-intentions,[2] ***Start***, ***Stop***, ***Choose Section*** and ***Enact Section***. The ***Start*** meta-intention starts the construction of a process by selecting a section in the method map which has map intention *Start* as source. The ***Choose Section*** meta-intention results in the selection of a

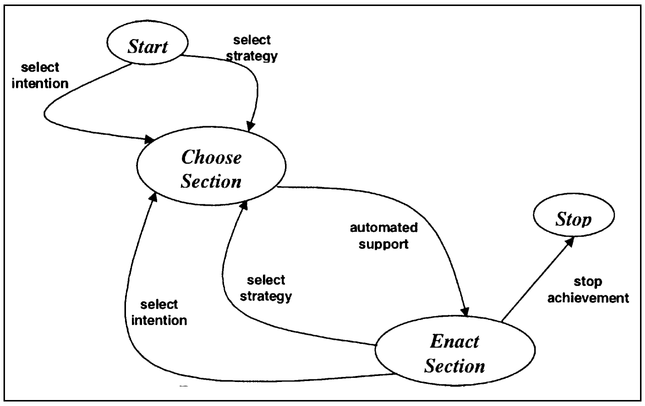


Fig. 13. Meta-process map.

method map section. The ***Enact Section*** meta-intention causes the execution of the method map section resulting from ***Choose Section***. Finally, the ***Stop*** meta-intention stops the construction of the application process. This happens when the ***Enact Section*** meta-intention leads to the enactment of the method map section having *Stop* as the target.

As already explained in the previous sections, there are two ways in which a section of a method map can be selected, namely by selecting an intention or by selecting a strategy. Therefore, the meta-intention ***Choose Section*** has two meta-strategies associated with it, **select intention** and **select strategy** respectively. Once a method map section has been selected by ***Choose Section***, the IAG to support its enactment must be retrieved; this is represented in Fig. 13 by associating the meta-strategy **automated support** with the meta-intention, ***Enact Section***.

When these meta-strategies are used together with the meta-intentions, then six sections as shown in the figure

[2]Meta-intentions and the meta-strategies are in bold but with the fonts used for the intentions and strategies (italics and Arial font respectively).

are formed. When progressing from ***Start*** to ***Choose Section*** the application engineer can use either **select intention** or **select strategy** depending on whether the intention of the application process is unknown or the intention is known but the strategy is unknown. A similar situation occurs when progressing from ***Enact Section*** to ***Choose Section***. There is only one strategy to proceed from ***Choose Section*** to ***Enact Section***, namely **automated support**. Similarly, when ***Choose section*** progresses to ***Stop*** then the **stop achievement strategy** is used.

There are three key meta-IAGs for achievement of the meta-intentions. These perform the selection of the guidelines of the method map:

- ISGs for *Choose section* with select intention
- SSGs for *Choose Section* with select strategy
- AGs for *Enact Section* with automated support

In the next section, we apply the meta-process model to generate a process which will produce the requirements specification of a recycling machine in a supermarket.

6. A Process for Eliciting Requirements of a Recycling Machine

This section illustrates the generation of a process for the Recycling Machine (RM) case study [15]. The initial situation is that of a supermarket wanting to provide recycling facilities to its customers. The map of the CREWS-L'Ecritoire (Cl) method presented in Fig. 9(a) is used by the meta-process to elicit the requirements of this machine. This method map will be referred to in the following as the Cl map.

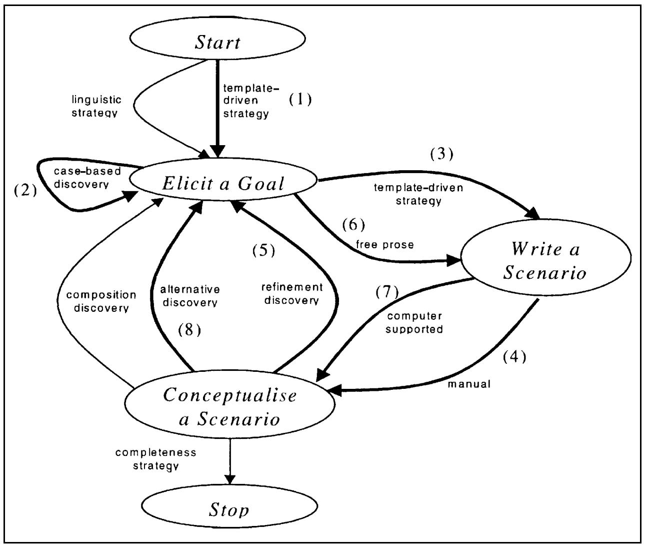


Fig. 14. Use of CL map for RM example.

The meta-process is used to drive the selection of the appropriate section in the CL map and to enact the CL guidelines in order to elicit the requirements for the RM. Figure 14 highlights the eight sections of the CL map selected and enacted as examples of the process steps for the RM. These sections are sequentially numbered according to the order in which they are selected and enacted.

Figure 15 shows the corresponding sequence of sections in the meta-map. Clearly, each step of the RM process results from two iterations in the meta-map: one to guide the selection of the appropriate section in the CL map for the situation at hand and the other one to guide the enactment of the IAG associated to the CL selected section (denoted n.1 and n.2 respectively for any step n in Fig. 15). The trace of the eight steps in both the meta-process and of the process is shown in Table 1. In the following we explain the interaction between the meta-process, the CL map and the requirements engineer for the first process step. The other steps will be interpreted from Table 1 in the same way.

The meta-process begins from the meta-intention ***Start***. In the Cl map there is exactly one intention, namely *Elicit a Goal* with *Start* as a source. Therefore, the meta-strategy is clearly **select strategy** to ***Choose Section*** (see Fig. 15). The achievement of the ***Choose Section*** following **select strategy** leads to the presentation of the SSG4 guideline (column 1 in the first row of Table 1) to the requirements engineer. The argument used by the requirements engineer to select from the choices offered by SSG4 is shown in the second column of the first row of Table 1. The result of this is the selected section shown in the third column of this row. This explains how the meta-map helps the requirements engineer selecting a section in the CL map. It is summarised in the first row of step 1 in Table 1.

Now, in the meta-process the next meta-intention is ***Enact Section*** (see Fig. 15), which is to be achieved by using the **automated support** meta-strategy. In the Cl

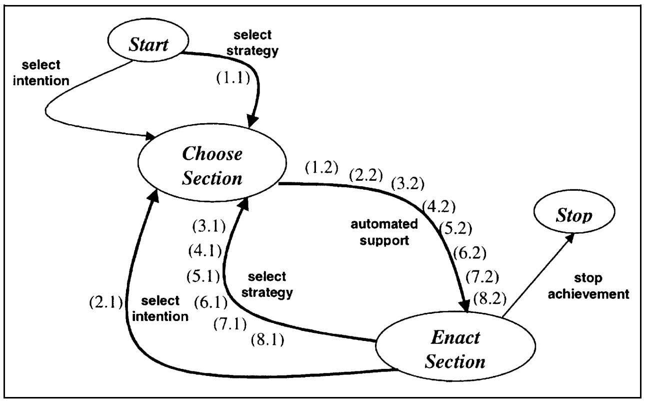
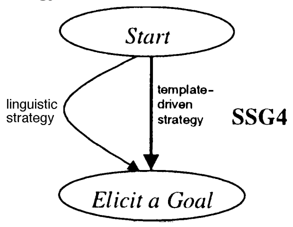
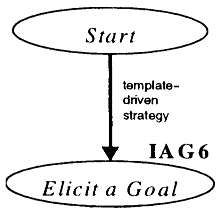
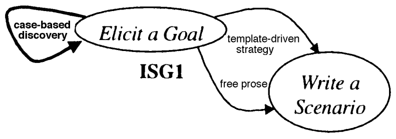
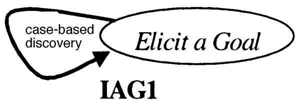


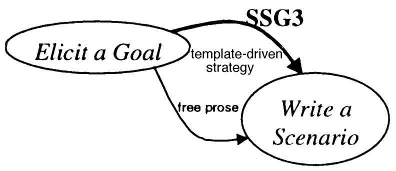
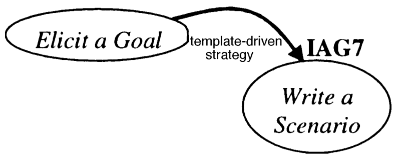
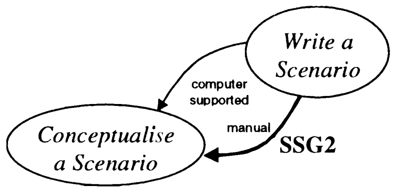
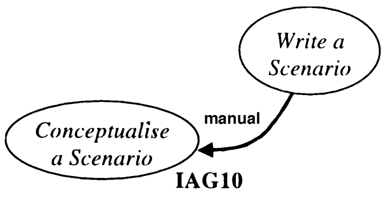
Fig. 15. Use of meta-process for RM example.

Table 1. Trace of the process to elicit requirements for the Recycling Machine case study

Step number	***Meta-process***	***Process***	
	Column 1 **Displayed guidelines**	***Column 2*** **IS & Guidelines Arguments**	***Column 3*** **Selected section**
1	Iteration 1.1 ***Choose section with select strategy***  Start linguistic strategy template-driven strategy SSG4 Elicit a Goal	SSG4 suggests two strategies. The template-driven strategy is chosen because it is the most appropriate way to become familiar with the goal formalisation proposed by the CREWS-L'Ecritoire method	(*Start*, *Elicit a Goal*, template-driven strategy)
		IA Guidelines Arguments	**Product**
	Iteration 1.2 ***Enact section with automated support*** 	IAG6 displays a goal statement template and explains the meaning of each parameter. The requirement Engineer (RE) chooses a loose statement having only a verb and a target	G1: Provide$_{verb}$ (Recylcing Facilities*)$_{target}$ *RF
	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
2	Iteration 2.1 ***Choose section with select intention***  case-based discovery Elicit a Goal template-driven strategy ISG1 free prose Write a Scenario	ISG1 provides RE with arguments to advise him on choosing one of the two possible intentions from *Elicit a Goal*, namely to *Elicit a Goal* or to *Write a Scenario*. The former is selected so as to generate alternative design solutions	(*Elicit a Goal*), *Elicit a Goal*, case-based strategy)
		IA Guidelines Arguments	**Product**
	Iteration 2.2 ***Enact section with automated support*** 	IAG1 uses the goal statement structure and parameter values supplied to generate alternative goals. This leads to 21 alternative goals to G1 which are ORed to G1. After discussion with stakeholders, G4 is selected	G2: Provide bottle RF to our customers with a card-based machine G3: Provide paper RF to our customers with a card-based machine G4: Provide bottle and box RR to our customers with a card-based machine . . . G22: Provide bottle RF to all customers with money return machine

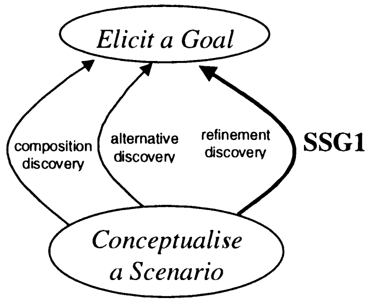
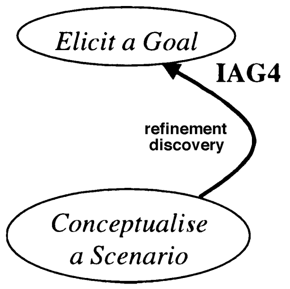
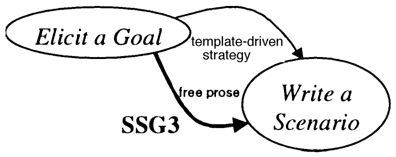
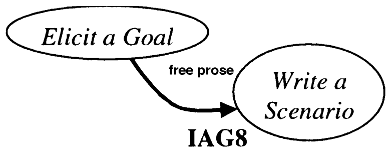
(*continued over*)

Table 1. Continued

Step number	***Meta-process*** ***Column 1*** ***Displayed guidelines***	***Process*** ***Column 2*** **IS & Guidelines Arguments**	***Column 3*** **Selected section**
	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
3	Iteration 3.1 ***Choose section with select strategy***  SSG3 Elicit a Goal template-driven strategy free prose Write a Scenario	SSG3 offers two strategies from which the template-driven strategy is chosen. This is because there is uncertainty about what a scenerio should be. The templates lead to some certainty	(*Elicit a Goal, Write a Scenario,* template-driven strategy)
		IA Guidelines Arguments	**Product**
	Iteration 3.2 ***Enact section with automated support***  Elicit a Goal template-driven strategy IAG7 Write a Scenario	IAG7 proposes a template to be filled in. The template corresponds to a service scenario and contains actions that express services expected from the system	SC4: If the customer gets a card, he recycles objects
	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
4	Iteration 4.1 ***Choose section with select strategy***  Write a Scenario computer supported manual SSG2 Conceptualise a Scenario	SSG2 offers two strategies to conceptualise a Scenario. Among the two strategies, manual and computer based, the former is chosen since the service scenario (SC4) is very simple and can be handled manually	(*Elicit a Goal, Conceptualise a Scenario,* manual)
		IA Guidelines Arguments	**Product**
	Iteration 4.2 ***Enact section with automated support***  Write a Scenario manual Conceptualise a Scenario IAG10	IAG10 suggests two things: (1) to avoid anaphoric references such as he, she, etc. (2) to express atomic actions in an explicit ordering (3) to avoid ambiguities The scenario is rewritten accordingly	SC4: 1. The customer gets a card 2. The customer recycles boxes and bottles

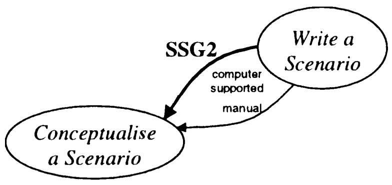
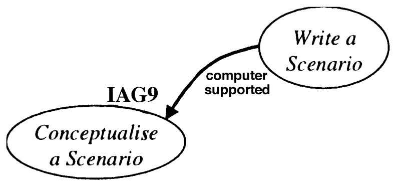
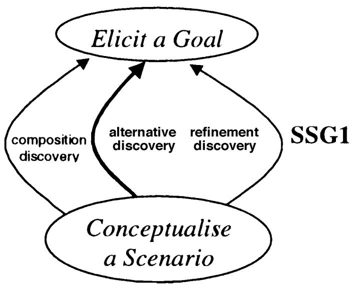
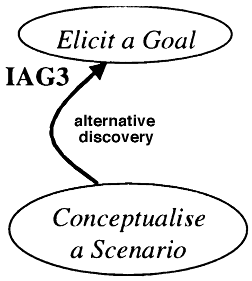
(*continued over*)

Table 1. Continued

	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
5	Iteration 5.1 ***Choose section with select strategy***  Elicit a Goal composition discovery alternative discovery refinement discovery SSG1 Conceptualise a Scenario	The RE knows that he wants to analyse the scenario SC4 to discover a new goal. Thus, he knows the target intention 'Elicit a Goal' and SSG1 is displayed. SSG1 offers three strategies to discover new goals from scenario analysis. The refinement strategy, is chosen because there is a need to discover the functional requirements of recycling machine	(*Conceptualise a Scenario, Elicit a Goal,* refinement discovery)
		IA Guidelines Arguments	**Product**
	Iteration 5.2 ***Enact section with automated support***  Elicit a Goal IAG4 refinement discovery Conceptualise a Scenario	IAG4 guides in transforming actions of the service scenario SC4 into goals which express functional requirements. Two goals are generated and related together to G4 with an AND relationship. G24 is selected for further processing	G23: Get card from supermarket G24: Recycle bottles and boxes from RM
	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
6	Iteration 6.1 ***Choose section with select strategy***  Elicit a Goal template-driven strategy free prose SSG3 Write a Scenario	The RE knows his target intention, namely 'Write a Scenario'. Thus SSG3 is displayed to help the RE in selecting the right strategy. The free prose strategy is selected because the text is likely to be long and the free prose facilitates this	(*Elicit a Goal, Write a Scenario,* free prose)
		IA Guidelines Arguments	**Product**
	Iteration 6.2 ***Enact section with automated support*** 	IAG8 provides style and contents guidelines adapted to the type of scenario at hand, namely system interaction scenario	SC24[1]: The customer inserts his card in the RM. The RM checks if the card is valid and then a prompt is given. The customer inputs the bottles and/or boxes in the RM. If the objects are not blocked, the RM ejects the card and prints a receipt

(*continued over*)

Table 1. Continued

	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
7	Iteration 7.1 ***Choose section with select strategy***  SSG2 Write a Scenario computer supported manual Conceptualise a Scenario	SSG2 is displayed. The automated support strategy is selected to take advantage of the powerful linguistic devices and obtain a scenario formulation which will be the basis for automated reasoning	(*Write a Scenario, Conceptualise a Scenario,* automated support)
		IA Guidelines Arguments	**Product**
	Iteration 7.2 ***Enact section with automated support***  Write a Scenario computer supported IAG9 Conceptualise a Scenario	IAG9 semi-automatically transforms the initial prose into a structured text whose semantics conform to the scenario model. The transformation includes disambiguation, completion and mapping onto the linguistic structures associated to the concepts of the scenario model. SC24^2 is the result of the transformation of SC24^1. (Underlined statements result of the transformation)	SC24^2: 1. The customer inserts the customer card in the RM 2. The RM checks if the card is valid 3. If the card is valid 4. A prompt is given to the customer 5. The customer inputs the bottles and the boxes in the RM 6. The RM checks if the bottles and the boxes are not blocked 7. If the bottles and the boxes are not blocked 8. The RM ejects the card to the customer 9. The RM prints a receipt to the customer
	Displayed guidelines	**IS & SS Guidelines Arguments**	**Selected section**
8	Iteration 8.1 ***Choose section with select strategy***  Elicit a Goal composition discovery alternative discovery refinement discovery SSG1 Conceptualise a Scenario	Of the three strategies proposed by SSG1, the alternative discovery strategy is chosen. This strategy suits the need to investigate variations and exceptions of the normal course of actions described in SC24^2	(*Conceptualise a Scenario, Elicit a Goal,* alternative discovery)
		IA Guidelines Arguments	**Product**
	Iteration 8.2 ***Enact section with automated support***  Elicit a Goal IAG3 alternative discovery Conceptualise a Scenario	IAG3 proposes several tactics to discover alternative goals to G24. The one based on the analysis of conditions in the scenario is selected. This leads to discover G25 and G26	G25: Recycle box and bottles from RM with invalid card G26: Recycle box and bottles with a deblocking phase

map this results in the selection of the IAG6 guideline that is displayed to the requirements engineer. This is shown in column 1 of the second row in Table 1. The enactment of this guideline is discussed in the second column of the second row of the table. The impact of this enactment on the product is shown in the last column of this row.

Thus the second row in Table 1 for a given step sums up the effect of enacting the IAG guideline corresponding to the section selected in the first row of the table for this step.

Now, in the meta-process, the next meta-intention is ***Choose Section*** with one of the two meta-strategies **select strategy** and **select intention**. This starts step 2 in the RM process. Since in the Cl map there are two intentions which can be achieved, the meta-strategy selected is **select intention** (see Fig. 15). As traced in the first column of the first row for step 2 in Table 1, this selection results in an achievement of the ***Choose Section*** leading to the presentation of the ISG1 guideline to the requirements engineer. The argument used by the requirements engineer is shown in the second column of this row of the table and the resulting selected section is shown in the last column.

In this way, the interaction of the meta-process, the Cl map and the application engineer continues. Eight iterations in the meta-process are shown in Table 1. These generate a partial specification of the RM.

The arguments contained in column 2 of the table show the use of non-determinism in intention and strategy selection embodied in the map. It also shows that for a given type of situation different strategies are chosen for different situations (instances) of this type. This effect is seen in iterations 3 and 6, 4 and 7 as well as in 5 and 8.

7. Conclusion

Early process models presented a take it or leave it choice to application engineers: either you adopted a certain model or you discarded it and chose another one. However, the recognition of the role of process situations in shaping the process model has resulted in adapting process models to situational needs. The basic approach to process modelling has, however, remained the same: process models are statically defined even though they are expected to handle dynamically changing situations. In other words, knowledge of all situations likely to occur is assumed to be statically available. This is clearly an untenable assumption.

Our approach is to respond to a dynamically changing situation by constructing process models dynamically. As a result, the process model handles a situation as it emerges and it is completely sensitive to the situation at all times.

Prevalent approaches to process modelling emphasise task organisation and are therefore principally concerned with the tactics to be adopted in carrying out the task. In the multi-model view presented here, we have called for a shift to the relatively more upstream activities performed to develop real processes: those of deciding what is to be done (intentions) and the manner (strategies) in which this is to be done. Thus, our focus is on strategic issues concerning process modelling. In fact, we separate the strategic from the tactical by representing the former in the method map and embodying the latter in the guidelines. By associating the guidelines with the map, a smooth integration of the strategic and the tactical aspects is achieved.

The capability to dynamically construct process models provided in the multi-model view is directly related to the identification of intentions and strategies needed. The dynamicity is promoted by the fine-grained modularity of sections and their high interconnectivity. This encourages flexible manoeuvrability in constructing multiple paths from the map.

Acknowledgements. This work was partly funded by the Basic Research Action CREWS (ESPRIT no. 21.903). CREWS stands for Cooperative Requirements Engineering With Scenarios.

References

1. Dowson M. Software process themes and issues. In: IEEE 2nd international conference on the software process, Berlin, Germany 1993, pp 28–40
2. Armenise P, Bandinelli S, Ghezzi C, Morzenti A. A survey and assessment of software process representation formalisms. Int J Software Eng Knowl Eng 1993;3(3):455–462
3. Jarke M, Pohl K, Rolland C, Schmitt JR. Experienced-based method evaluation and improvement: a process modeling approach. In: International IFIP WG 8.1 conference in CRIS series: method and associated tools for the information systems life cycle. North-Holland, Amsterdam, 1994
4. Finkelstein A, Kramer J, Nuseibeh B (eds). Software process modelling and technology. Wiley, New York, 1994
5. Hidding GJ. Methodology information: who uses it and why not? In: Proceedings of WITS-94, Vancouver, Canada, 1994
6. Russo. The use and adaptation of system development methodologies. In: Proceedings of the 1995 international Resources Management Association conference, Atlanta, GA, 1995
7. Wijers GM, van Dort HE. Experiences with the use of CASE tools in the Netherlands. Adv Inform Syst Eng 1990;5–20
8. Aaen et al. A tale of two countries: CASE experience and expectations. The impact of computer supported technology on information systems development. North-Holland, Amsterdam, 1992, pp 61–93
9. Yourdon E. The decline and fall of the American programmer. Prentice-Hall, Englewood Cliffs, NJ, 1992
10. Cugola G, Di Nitto E, Ghezzi C, Mantione M. How to deal with deviations during process model enactment. In: Proceedings of the 17th international conference on ‘software engineering’ (ICSE17), Seattle, WA, April 1995

11. Cugola G, Di Nitto E, Fuggetta A, Ghezzi C. A framework for formalizing inconsistencies and deviations in human-centered systems. ACM Trans Software Eng Methodol 1996;5(3)191–230
12. Cugola G. Inconsistencies and deviations in process support systems. PhD thesis, Politecnico di Milano, February 1998
13. Dowson M, Fernstrom C. Towards requirements for enactment mechanisms. In: Proceedings of the 3rd European workshop on ‘software process technology’, France, 1994
14. Si-Said S, Rolland C, Grosz G. MENTOR: a computer aided requirements engineering environment. In: Proceedings of CAiSE ’96, Crete, May, 1996
15. Jacobson I, Christerson M, Jonsson P, Oevergaard G. Object oriented software engineering: a use case driven approach. Addison-Wesley, Reading, MA, 1992
16. Bandinelli S, Fugetta A, Grigoli S. Process modelling in the large with SLANG. In: Proceedings of the 2nd international conference on ‘software process’, Berlin, 1993, pp 75–93
17. Belkhatir N, Melo WL. Supporting software development processes in Adele2. Comput J 1994;37(7):621–628
18. Welke RJ, Kumar K., Method engineering: a proposal for situation-specific methodology construction. In: Cotterman and Senn (eds). Systems analysis and design: a research agenda. Wiley, New York, 1992, pp 257–268
19. Harmsen AF, Brinkkemper JN, Oei JLH. Situational method engineering for information systems project approaches. In: International IFIP WG 8.1 conference in CRIS series: methods and associated tools for the information systems life cycle (A-55). North-Holland, Amsterdam, 1994
20. Van Slooten K, Hodes B. Characterising IS development project. In: IFIP WG 8.1 conference on method engineering. Chapman & Hall, London, 1996, pp 29–44
21. Euromethod VI. Euromethod project, final deliverable, 1996
22. Ghannouchi SA, Ben Ghesala HH. Utilisation du méta-modèle NATURE pour modéliser le processus de retro-conception de données. In: Proceedings of the 1st international workshop on the ‘many facets of process engineering’, Gammarth, Tunisia, 22–23 September 1997
23. Rolland C, Souveyet C, Moreno M. An approach for defining ways-of-working. Inform Syst J 1995;20(4)337–359
24. Pohl K. PRO-ART: an environment for enabling requirements traceability. In: Proceedings of the 2nd international conference on ‘requirements engineering’, Colorado Springs, CO, 1996
25. Bubenko J, Rolland C, Loucopoulos P, De Antonellis V. Facilitating ‘fuzzy to formal’ requirements modelling. In: IEEE 1st conference on ‘requirements engineering’ (ICRE ’94), 1994, pp 154–158
26. Plihon V. The OMT, OOA, SA/SD, E/R, O*, OOD methodologies, NATURE Deliverable DP2, 1994
27. Prat N. Goal formalisation and classification for requirements engineering. In: Proceedings of the third international workshop on ‘requirements engineering: foundations of software quality’ (REFSQ ’97), Barcelona, June 1997, pp 145–156
28. Rolland C, Souveyet C, Ben Achour C. Guiding goal modelling using scenarios. IEEE Trans Software Eng (Special Issue on Scenario Management) 1998;24(12):1055–1071
29. Le Petit Robert French Dictionary, Dictionnaires Le Robert, France, 1995
30. Grosz G, Rolland C, Schwer S et al. Modelling and engineering the requirements engineering process: an overview of the NATURE approach. Requirements Eng J 1997;2:115–131
31. Si Said S. Guidance for requirements engineering processes. In: Proceedings of the 8th international conference and workshop on ‘database and experts system application’, DEXA’97, Toulouse, 1–5 September 1997
32. Rolland C. Modelling the requirements engineering process. In: 3rd European–Japanese seminar on ‘information modelling and knowledge bases’, Budapest, June 1993
33. Rolland C, Grosz G. A general framework for describing the requirements engineering process. In: IEEE conference on ‘systems, man and cybernetics’ (CSMC94), San Antonio, TX, 1994
34. Rolland C, Prakash N. Guiding the requirements engineering process. In: Proceedings of the IEEE Asia–Pacific software engineering conference (APSEC), Tokyo, 1994
35. Sutcliffe AG, Jarke M, Rolland C, Bubenko J, Constantopoulos P. Defining visions in context models, process and tools for requirements engineering. Inform Syst J 1997;21(6):515–547
36. Ben Achour C. Guiding scenario authoring. In: Proceedings of the 8th European–Japanese conference on ‘information modelling and knowledge bases’, Ellivuori, Finland, 26–29 May 1998, pp 181–200
37. Rolland C, Ben Achour C. Guiding the construction of textual use case specifications. Data Knowl Eng J 1997;25(1):125–160
38. Haumer P, Pohl K, Weidenhaupt K. Requirements elicitation and validation with real world scenes. IEEE Trans Software Eng (Special Issue on Scenario Management) 1998;24(12)
39. Sutcliffe AG, Maiden NAM, Minocha S, Manuel D. Supporting scenario-based requirements engineering. Trans Software Eng (Special Issue on Scenario Management) 1998
40. Dubois E, Heymans P. Scenario-based techniques for supporting the elaboration and the validation of formal requirements. Requirement Eng J 1998;3(3–4):202–218
41. Ralyté J, Rolland C, Plihon V. Method enhancement by scenario based techniques. In: Proceedings of the 11th conference on advanced information systems engineering, Heidelberg, Germany, June 1999
42. www.univ-paris1.fr/CRINFO/METHODBASE/
43. Nurcan S, Rolland C. 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
44. Rolland C. A comprehensive view of process engineering. In: Pernici, Thanos C (eds). Proceedings of the 10th international conference, CAiSE ’98, B. Lecture notes in computer science 1413, Pisa, June 1998. Springer, Berlin