Guiding Goal Modeling Using Scenarios

Colette Rolland, Carine Souveyet, and Camille Ben Achour

Abstract—Even though goal modeling is an effective approach to requirements engineering, it is known to present a number of difficulties in practice. The paper discusses these difficulties and proposes to couple goal modeling and scenario authoring to overcome them. Whereas existing techniques use scenarios to concretize goals, we use them to discover goals. Our proposal is to define enactable rules which form the basis of a software environment called L'Ecritoire to guide the requirements elicitation process through interleaved goal modeling and scenario authoring. The focus of the paper is on the discovery of goals from scenarios. The discovery process is centered around the notion of a requirement chunk (RC) which is a pair <Goal, Scenario>. The paper presents the notion of RC, the rules to support the discovery of RCs and illustrates the application of the approach within L'Ecritoire using the ATM example. It also evaluates the potential practical benefits expected from the use of the approach.

Index Terms—Goal modeling, scenario authoring, goal discovery.

1 Introduction

GOAL MODELING is an effective way to identify requirements [25]. The argument of goal driven approaches is that the rationale for developing a system is to be found outside the system itself, in the enterprise [19] in which the system shall function. We have applied the goal driven approach as embodied in the EKD method [4], [17], [20], [30] to several domains, air traffic control, electricity supply, human resource management, and tool set development. Our experience is that it is difficult for domain experts to deal with the fuzzy concept of a goal. Yet, domain experts need to discover the goals of real systems. It is often assumed that systems are constructed with some goals in mind [8]. However, practical experience [2], [9] shows that goals are not given and therefore the question as to where they originate from [2] acquires importance. In addition, enterprise goals which initiate the goal process do not reflect the actual situation but an idealized environmental one. Therefore, proceeding from this may lead to ineffective requirements. Thus, goal discovery is rarely an easy task. Additionally, it has been shown [2] that the application of goal reduction methods [7], to discover the component goals of a goal, is not as straightforward as the literature suggests. Our own experience in the F^3 [4] and ELEKTRA [10] projects is also similar. It is thus evident that help has to be provided so that goal modeling can be meaningfully performed. This help must: 1) facilitate the work of the domain expert by getting over the problem of the fuzzy nature of goals, 2) help discover goals, and 3) aid in the task of goal reduction.

Independently of goal modeling, an alternative approach to requirements engineering, the *scenario-based approach*, has been proposed. By capturing examples and illustrations, scenarios help people in reasoning about complex systems [24]. Since scenarios describe real situations, they capture real requirements. However, because they deal with examples and illustrations, scenarios only provide restricted requirements descriptions which need to be generalized to obtain complete requirements.

Recently, some proposals have been made to *couple goals and scenarios* together. Potts [25] claims that it is «unwise to apply goal based requirements methods in isolation» and suggests that they should be complemented with scenarios. However, he does not make a specific proposal on how this can be done. Yet other proposals exist which interpret scenarios as containing information on how goals can be achieved [1], [14], [25]. Thus, the goal-scenario combination has been used to operationalize goals. Yet others look upon goals as playing a documentation role only. This view is taken in [6], [15], [18], [23] where a goal is considered as a contextual property of a use case (integrated set of scenarios) i.e., it is a property that relates the scenario to its organizational context. Cockburn [5] goes beyond this view and suggests the use of goals to structure use cases by connecting every action in a scenario to a goal assigned to an actor. In this sense a scenario is discovered each time a goal is.

All these views suggest a *unidirectional relationship* between goals and scenarios (goal operationalization through scenarios, scenario discovery from goals). Further, they contain no proposal to formally track goals nor are there any methodological guidelines on how to associate goals with scenarios. Finally, these approaches do not directly help in discovering/clarifying new requirements. Thus, they tackle the problems of goal achievement rather than goal discovery, documentation rather than requirements description, and scenario structuring rather than requirements elaboration, respectively.

In the ESPRIT CREWS project, we propose to couple goals and scenarios to directly help in the requirements engineering activity. Thus, our aim is to discover/elicit requirements through goal-scenario coupling. Three characteristics of the proposed approach contribute to the achievement of this objective:

• *C. Rolland, C. Souveyet, and C. Ben Achour are with the Centre de Recherche en Informatique, Université de Paris-1 Sorbonne 17, rue de la Sorbonne, 75231 Paris cedex 05, France.*
E-mail: {rolland,souveyet,camille}@univ-paris1.fr.

Manuscript received 5 Jan. 1998; revised 3 July 1998.
Recommended for acceptance by R. Kurki-Suonio and M. Jarke.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 107461.

First, a *bidirectional goal-scenario coupling:* just as goals can help in scenario discovery, so also scenarios can help in goal discovery. Thus, the requirements elicitation process can be organized in two phases: 1) *scenario authoring* and 2) *goal discovery* (see Fig. 1). As each individual goal is discovered, a scenario can be authored for it. In this sense, the goal-scenario coupling is exploited in the forward direction, from goals to scenarios. Once a scenario has been authored, it is explored to yield goals. This leads to goal discovery by moving along the goal-scenario relationship in the reverse direction. The problem of goal discovery mentioned at 2) is, therefore, tackled here through scenario authoring and subsequent goal identification.

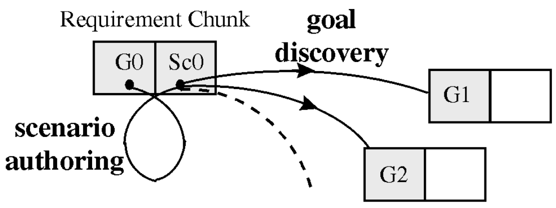


Fig. 1. Overview of the requirements elicitation process.

The second characteristic of the approach is the distinction between the *refinement relationship* and the *AND/OR relationships* among goals. This leads to an organization of the collection of requirements as a hierarchy of Requirement Chunks (RCs) related through AND, OR, and Refinement relationships. A *requirement chunk* (RC) is defined as a pair <G, Sc> where G is a goal, and Sc is a scenario. The RCs and their inter-relationships constitute a system of concepts which we call the *requirement chunk model* (the RC model). We propose the refinement relationship under which starting from fuzzy goals more concrete goals are discovered. Each of these concrete goals has its own AND/OR goal hierarchy. Therefore, the problems 1) and 3) with goal modeling mentioned above are addressed: the refinement relationship ensures that fuzzy goals are gradually made more and more clear; goal reduction is possible through traditional AND/OR structuring of goals.

The third characteristic is the *methodological support* provided in the form of *enactable guiding rules* embodied in a computer software environment called *L'Ecritoire*. As a result, it is possible to guide the requirements elicitation process through interleaved goal modeling and scenario authoring. Whereas rules for scenario authoring can be found in [3], [8], we propose here a basis for guidance in goal modeling. There are three kinds of rules, each for determining one of the three kinds of relationships mentioned above namely, AND, OR, and refinement. Each rule generates a menu of goals for the requirement chunk Author (RCA) to evaluate and select. In this way, the drudgery of goal-menu generation is removed and the engineer has only to concentrate on the more intellectually demanding task of goal evaluation and selection.

The *Crews-L'Ecritoire* approach has been evaluated and improved through experience gained in the F^3 [4] and ELEKTRA [10] European ESPRIT projects. The latter contained as many as 300 RCs. We are now involved in an evaluation of the approach with industrial partners and in scaling up the software tool *L'Ecritoire* to a full working system. To study stakeholders' reaction, three one-day workshops on the *Crews-L'Ecritoire* approach were held. These were attended by 52 participants. The overall evaluation of the main aspects of the approach is encouraging and highlights the four following potential practical benefits:

- methodological support,
- tight coupling of requirements and scenarios,
- requirements tracking from high to low level, and
- guided mapping from informal to formal scenario descriptions.

This feedback from participants is reported in Section 5.

The layout of the paper is as follows. The RC model is described in the next section. The three kinds of rules and the discovery process are presented in Section 3. The rules are applied to the CREWS goal-oriented approach through a *Crews-L'Ecritoire* session in Section 4. Section 5 deals with the potential of the approach to overcome some of the known industrial problems and reports on the industrial workshops. The concluding section sums up the essential properties of our approach, and shows how it alleviates the problems of goal modeling.

2 THE REQUIREMENT CHUNK MODEL

At the core of our approach is the requirement chunk (RC), defined as the pair <Goal, Scenario>. Requirement chunks can be assembled together either through *composition* and *alternative* relationships or through *refinement* relationships. The former lead to AND and OR structure of RCs whereas the latter leads to the organization of the RCs as a hierarchy of chunks at different levels of abstraction. Fig. 2 gives, in the OMT notation [31], an overview of the RC model.

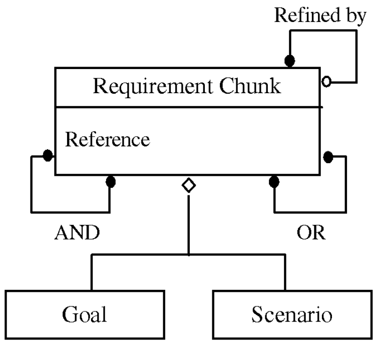


Fig. 2. Overview of the requirement chunk model.

2.1 The Requirement Chunk

The requirement chunk (RC) is the basic building block of the requirement chunk model. A requirement chunk 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 by nature, a requirement chunk is a possible way of achieving the goal. We model the requirement chunk (Fig. 2) as a class of objects which is an aggregate of the goal and scenario classes. Both goal and scenario are complex objects which are explained in the following sections.

2.1.1 The Goal Concept

A *goal* is defined [22] as "something that some stakeholder hopes to achieve in the future." The structure of a goal is shown in Fig. 3. Clearly, a goal [26] is associated to a verb and to one or more parameters (multiplicity is shown by a black dot). It is expressed as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb. There are four types of parameters (shown in the grey boxes), some of which have subtypes. These subtypes are described and illustrated with the ATM example.

The *target* (Tar) designates entities affected by the goal. We distinguish two types of targets, object, and results. An *object* (Obj) is supposed to exist before the goal is achieved. For example in the goal:

'*Take (the receipt)$_{Obj}$ (from the printer)$_{So}$*',

the target 'the receipt' is an object because it exists even before '*Take*' is achieved. *Results* (Res) can be of two kinds 1) entities which do not exist before the goal is achieved and 2) abstract entities which exist but are made concrete as a result of goal achievement. For example in the goal statement:

'*Identify (the user's choice)$_{Res}$*',

the user's choice is the result of the achievement of the goal '*Identify*'.

The two types of *direction* (Dir), namely s*ource* (So) and *destination* (Dest) identify respectively the initial and final location of objects to be communicated. For example, consider the two goals as follows:

'*Read (the validity date of card)$_{Obj}$ (in the card chip)$_{So}$*,'
'*Display (the error message)$_{Obj}$ (to the customer)$_{Dest}$*,'

In the first goal, the source of the validity date of the card is the card chip, and in the second one, the customer is the destination of the error message.

Means (Mea) designate entities which act as instruments using which a goal is to be performed. For example, given the goal:

'*Provide (cash)$_{Obj}$ (to our bank customers)$_{Dest}$ (with a finger print based ATM)$_{Mea}$*',

the *finger print based ATM* is the means to provide cash.

The *manner* (Man) defines the way in which the goal is to be achieved. As modeled in Fig. 3 a manner, when complex, can be expressed as a goal. Therefore, a goal can be recursively defined. For example the goal statement:

'*Improve (our services)$_{Obj}$ (by providing (cash)$_{Obj}$ (to our bank customers)$_{Dest}$ (from account)$_{So}$(with a card based ATM)$_{Mea}$)$_{Man}$*',

comprises a recursive definition of the manner 'by providing (cash)$_{Obj}$ (to our bank customers)$_{Dest}$ (from account)$_{So}$ (with a card based ATM)$_{Mea}$' which is itself a goal comprising the verb 'provide' and four parameters.

The *beneficiary* (Ben) is the person (or group of persons) in favor of whom the goal is to be achieved; for example in:

'*Reduce (the work load)$_{Obj}$ (for the bank staff)$_{Ben}$*',

the 'bank staff' is the beneficiary.

2.1.2 The Scenario Concept

A *scenario* is "a possible behavior limited to a set of purposeful interactions taking place among several agents" [22]. Fig. 4 shows that a scenario is composed of one or several *actions*, the combination of actions in a scenario describes a unique path leading from initial to final states of agents. Thus, it is the combination of scenarios that describes the behavior of a complex system of agents. We are aware that not all the possible behaviors can be expressed through combinations of scenarios. However, as [5], [27], among others, we advocate that they are sufficient to express a majority of the behaviors that are necessary for the purpose of scenario based goal modeling.

A scenario is characterized by initial and final states. An *initial state* attached to a scenario defines a precondition for the scenario to be triggered. For example, the scenario '*Withdraw cash from ATM in a normal way*' cannot be performed if the initial states '*The user has a card*' and '*The ATM is ready*' are not true. A *final state* defines a state reached at the end of the scenario. For example, the scenario '*Withdraw cash from ATM in a normal way*' leads to the states '*The user has cash*' and '*The ATM is ready*'.

As shown in Fig. 4 by the scenario class subtyping, we distinguish between *normal* and *exceptional* scenarios. The former leads to the achievement of its associated goal whereas the latter fails in goal achievement. The scenario

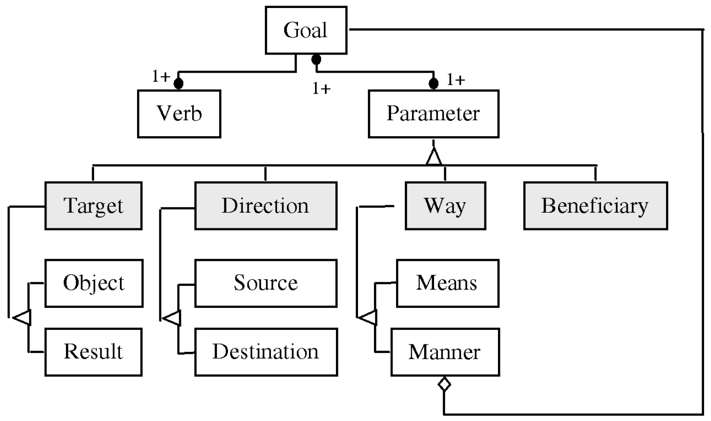


Fig. 3. The goal structure.

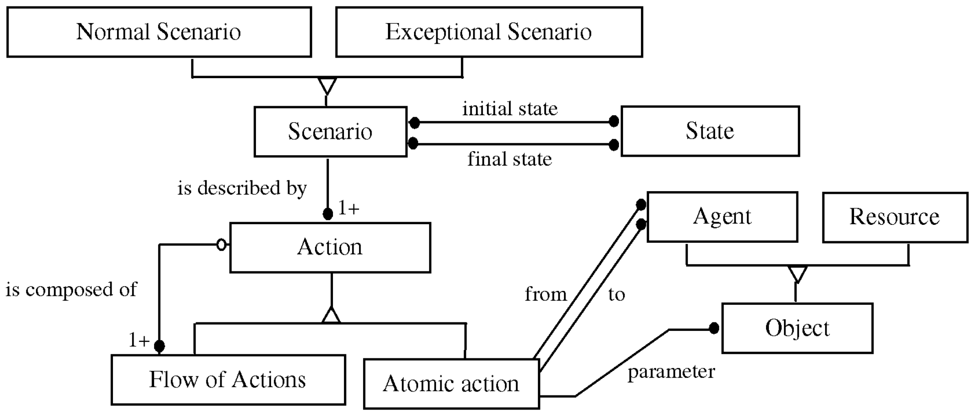


Fig. 4. The scenario structure.

'*Withdraw cash from the ATM by treating the exception of three invalid code attempts'* is an example of an exceptional scenario with a final state *'The user has no cash'*.

Actions are of two types: atomic and flows of actions. *Atomic actions* are interactions *from* one agent *to* another which affect some *parameter object* (see Fig. 4). An agent and resource objects may participate into several atomic actions. The clause *'The user inserts a card in the ATM'* is an example of atomic action. Its parameter, *'a card'* is a resource object, and it is a communication action which involves two different agents *'The user'* and *'the ATM'*.

Flows of actions are composed of several actions. The sentence *'The bank customer gets a card from the bank, then the bank customer withdraws cash from the ATM'* is an example of a flow of actions comprising two atomic actions. A flow of actions can have any one of the following semantics: sequence, alternative, repetition, and concurrency.

Alternative and repetition carry *flow conditions* which characterise the course of actions of the scenario. In the sentence *'if the code is valid, then a prompt for amount is displayed by the ATM to the use'*, the flow condition *'if the code is valid'* identifies a unique case of ATM usage which is described in the scenario.

In this paper, we shall use the semistructured textual form to represent the scenarios associated to requirement chunks. As shown in Fig. 5, each action of a scenario is expressed in a separate line as a natural language clause preceded by a reference number. The formal semantics of natural language clauses is detailed in [3], [28].

2.2 The Hierarchy of Requirement Chunks

The three types of relationships among requirement chunks lead to a hierarchical organization of RCs, as illustrated in Fig. 5.

2.2.1 Composition and Alternative Relationships

The composition and alternative relationships lead to a horizontal AND/OR structure between RCs. These are extensions of AND/OR relationships between goals identified by a number of researchers, for example in NATURE process theory [13], KAOS [7], F^3 [4] and others (e.g., [1], [11], [34]).

AND relationships among RCs (the OMT 'AND' association in Fig. 2) link together those chunks that require each other to define a completely functioning system. The requirement chunks RC1.1 and RC1.2 in Fig. 5 associated with the goals, G1.1 and G1.2 are examples of such chunks. Indeed, in order to *'Withdraw cash from ATM in a normal way'* (G1.1) it is necessary for the bank customer to *'Get a card from the bank'* (G1.2).

RCs related through *OR relationships* (the OMT 'OR' association in Fig. 2) represent alternative ways of fulfilling the same goal. There are, for example, at least three different manners to *'Withdraw cash from ATM'* namely, 1) *'in a normal way,'* 2) *'by treating the exception 'Invalid card'*, and 3) *'in a normal way with code error correction'*. These three manners[1] are captured in three requirement chunks (RC1.1, $RC.1.1^1$, and $RC1.1^2$ in Fig. 5) where the goals correspond to the same intention (*'Withdraw cash from ATM'*) but each having a different manner.

2.2.2 Refinement Relationship

Abstraction is defined [34] as a mechanism to hide details in order to focus on essential aspects. Refinement is used in our approach to describe requirement chunks at different levels of abstraction. This is modeled in Fig. 2 by the OMT association *'Refined by'*. As illustrated with the double arrows in Fig. 5, a RC at level i is refined into several RCs at level i + 1.

Refinement is directed by the scenario part of the requirement chunk. Every interaction in a scenario Sc at level i is looked upon as a goal to be achieved at level i + 1. For example, the chunk RC1 presented in Fig. 5, is refined by three requirement chunks[2] RC1.i, one for each action in the scenario Sc1. The refinement relationship establishes a vertical link between requirement chunks whereas the AND/OR relationship establishes a horizontal link between them.

To conclude, requirements engineering involves the creation and criticism of many descriptions of hypothetical system properties or environmental possibilities. These descriptions are captured in requirement chunks.

1. Manner is a subtype of way (see Fig. 3).

2. The notation of RC indices is the following: given a requirement chunk RCi, the indice i is incremented for ANDed RCs, an exponent is incremented for ORed RCs (e.g., RCi^1, RCi^2, RCi^3, etc.), and the dotted notation RCi.1 is used to refine RCs. Thus, the requirement chunks ANDed to RCi.1 are RCi.2, RCi.3, etc. However adequate for top down approaches of goal discovery, this numbering scheme is not adapted for supergoal discovery which is not addressed in this paper.

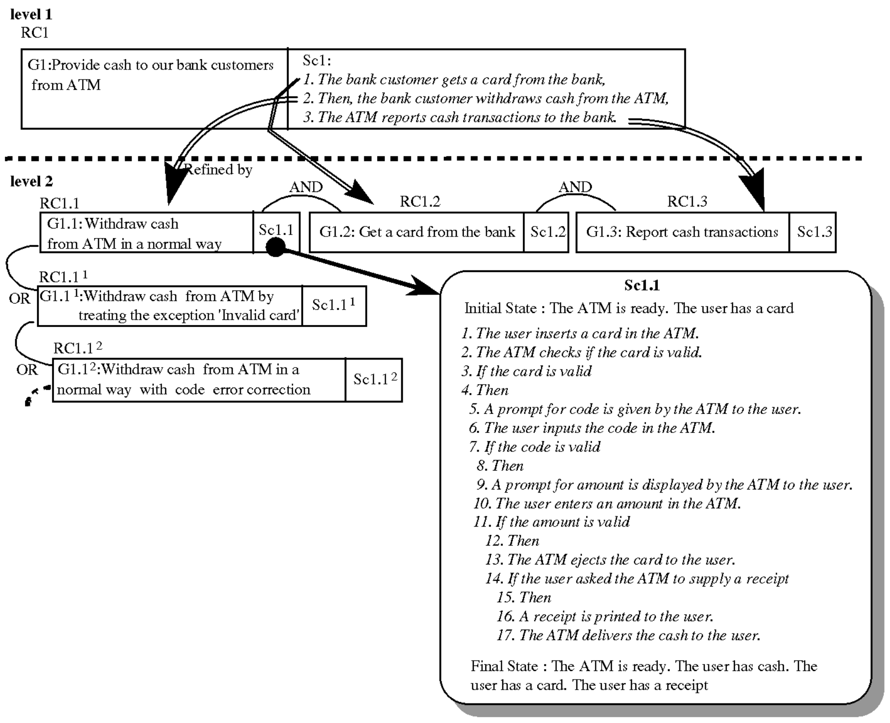

level 1
RC1
G1:Provide cash to our bank customers from ATM
Sc1:
1. The bank customer gets a card from the bank,
2. Then, the bank customer withdraws cash from the ATM,
3. The ATM reports cash transactions to the bank.
level 2
Refined by
AND
AND
RC1.1
G1.1: Withdraw cash from ATM in a normal way
Sc1.1
RC1.2
G1.2: Get a card from the bank
Sc1.2
RC1.3
G1.3: Report cash transactions
Sc1.3
RC1.1 1
OR
G1.1 1:Withdraw cash from ATM by treating the exception 'Invalid card'
Sc1.1 1
RC1.1 2
OR
G1.1 2:Withdraw cash from ATM in a normal way with code error correction
Sc1.1 2
Sc1.1
Initial State : The ATM is ready. The user has a card
1. The user inserts a card in the ATM.
2. The ATM checks if the card is valid.
3. If the card is valid
4. Then
5. A prompt for code is given by the ATM to the user.
6. The user inputs the code in the ATM.
7. If the code is valid
8. Then
9. A prompt for amount is displayed by the ATM to the user.
10. The user enters an amount in the ATM.
11. If the amount is valid
12. Then
13. The ATM ejects the card to the user.
14. If the user asked the ATM to supply a receipt
15. Then
16. A receipt is printed to the user.
17. The ATM delivers the cash to the user.
Final State : The ATM is ready. The user has cash. The user has a card. The user has a receipt

Fig. 5. An example of RCs hierarchy.

The alternative and composition relationships help in structuring these descriptions by separating chunks which represent alternatives of one another and chunks which are complementary to one another. The refinement relationship helps in moving from business goals through different levels of abstraction at the discretion of the requirement chunk author. In Section 4, we shall show the manner in which the different levels defined in the CREWS project [29], namely the *contextual*, *system interaction,* and *system internal* levels, are realized by the abstraction relationship. Distinguishing between the three kinds of relationships does not represent an attempt at modifying the traditional AND/OR structure of goal hierarchies. However, it does introduce goals at different abstraction levels as an additional feature in such a hierarchy.

The discovery of RCs' and relationships between them are both supported in our approach by guiding rules thus leading to a more systematic exploration of choices and a better support for requirements completeness and refinement. These rules are presented in the next section.

3 Discovering Goals

3.1 Overview of the Goal Discovery Process

In the requirements elicitation process, goal discovery and scenario authoring are complementary activities. Once a goal is discovered, scenario authoring can be done, followed by goal discovery. These goal-discovery/scenario-authoring sequence is repeated to incrementally populate the requirement chunks hierarchy (Fig. 6).

The requirements elicitation process can be viewed as a *flow of steps*: each step starts with a given goal and describes a scenario as a possible concretization of the goal. Clearly, a step results in a complete requirement chunk. In order to progress, a new goal has to be determined. This is done in the goal discovery activity through an analysis of the scenario. Thus, it can be said that the discovery activities regulate the flow of authoring activities.

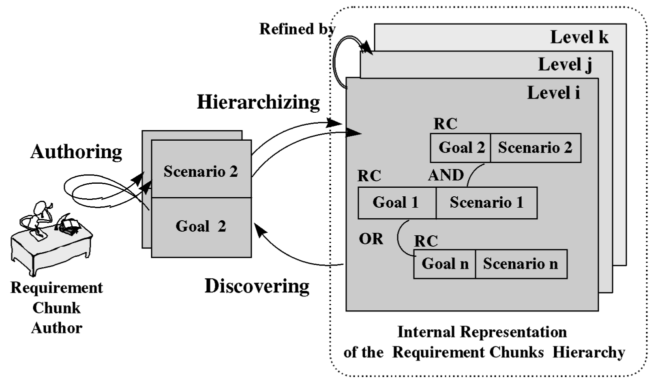


Fig. 6. Overview of the discovery process.

It has been shown that flow control is based on *strategies* [7], [24], [32]. We identify three strategies, namely *refinement*, *composition,* and *alternative* strategies. Upon the completion of a step, any strategy can be dynamically chosen. Thus, there is no statically imposed linear order of the flow of steps. This flexibility in strategy selection is the main advantage of the step-flow process. However, it shall be noticed that goal discovery in this approach is about subgoal discovery, not supergoal discovery.

The three discovery strategies exploit the three types of relationships identified among requirement chunks in the previous section. Therefore, given a pair <G, Sc>:

- the composition strategy looks for goals Gi which are ANDed to G,
- the alternative strategy searches for goals Gj which are ORed to G,

- the refinement strategy aims to discover goals Gk at a lower level of abstraction than G.

Thus, when a step is completed and the requirement chunk is expressed, the Requirement Chunk Author (RCA) has three options to proceed in the process.

We associate guiding rules (Fig. 7) with each of the strategies to discover new goals. Therefore, *composition* (*alternative*) *rules* help in discovering goals ANDed (ORed) to G. All these goals are at the same level of abstraction. The <G, Sc> chunk is processed by the *refinement rules* to provide goals at a lower level of abstraction than G. This is done by considering each interaction in Sc as a goal. Thus as many goals are produced as there are interactions in Sc.

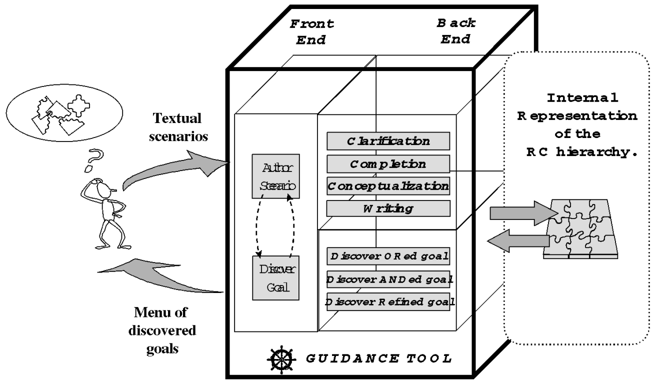


Fig. 7. Overview of *L'Ecritoire* software environment.

We have replaced the AND/OR decomposition used in goal modeling by refinement, composition and alternative strategies. The major contribution of our approach is that whereas decomposition is an ad hoc process, refinement can be systematically applied. This systematic application is possible, thanks to the scenario part of the requirement chunk. The advantages flowing from the refinement strategy and subsequent rules will be further elaborated in Section 5.

As shown in Fig. 7, the rules are implemented in the *L'Ecritoire software environment* and automatically enacted on request. This facilitates the use of the approach and limits the effort required to apply it. Using *L'Ecritoire*, the RCA is automatically guided in a flexible manner to elicit requirements systematically.

3.2 The Discovery Guiding Rules

The current corpus of rules comprises six guiding rules, two for each strategy. We developed a domain analysis approach to identify common structuring patterns and their underlying discriminent criteria in existing collections of scenarios. The formalization of these patterns results in the current set of guiding rules. The rules capture generic laws governing the construction of large collections of goals and scenarios. These laws are generic in the sense that they can be applied to the construction of many <Goal, Scenario> structures. Furthermore, the same rule can be applied at different levels of abstraction. The current corpus shall be extended as the domain analysis proceeds.

In this section, each rule is introduced using the following template <Goal, Body, Comment>. The *goal* is expressed in the notation given in Section 2. The *body* is expressed as a sequence of steps to be followed when applying the rule. The *comment* explains the rule. A formal description of the algorithmic parts of rules is presented in Appendix A.

Alternative guiding rule (A1)

Goal:

Discover (from goal G)$_{So}$ (goals ORed to G)$_{Res}$
(in a goal structure driven manner)$_{Man}$

Body:

Step1: Requirement Chunk Author (RCA) rephrases goal G according to the goal template,
Step2: RCA provides alternative parameters of goal G,
Step3: Compute all possible combinations of parameters,
Step4: Present the possible combinations to the RCA,
Step5: RCA evaluates and selects the goals of interest,
Step6: Requirement chunks corresponding to the selected goals are ORed with each other.

Comment:

The guiding rule A1 uses the goal G of a requirement chunk (*from goal G*)$_{So}$, to discover ORed goals to G (*goals ORed to G*)$_{Res}$. The discovery strategy (*in a goal structure driven manner*)$_{Man}$ exploits the goal structure defined in Section 2. First, the RCA is asked to identify the parameters of the goal. For example, the goal *'Provide cash to our bank customers from ATM'* will be restructured as follows:

'Provide (cash)$_{Res}$ (to our bank customers)$_{Dest}$ (from account)$_{So}$ (with a card based ATM)$_{Mea}$'

This leads in the above example, to introduce the source (*from account*)$_{So}$ and to specialize the means (*with a card based ATM*)$_{Mea}$. Then, the RCA identifies relevant alternative values to consider for each parameter (step 2). All possible combinations (step 3) of parameters with respect to the alternative values provided by the RCA are presented to him/her (step 4) as a list of possible ORed goals to the initial goal G. The RCA evaluates the proposals and selects the goals of interest (step 5).

Alternative guiding rule (A2)

Goal:

Discover (from requirement chunk <G, Sc>)$_{So}$
(goals ORed to G)$_{Res}$
(reasoning on flow conditions of Sc)$_{Man}$

Body:

Step1: Scan scenario description to construct the graph of paths of actions,
Step2: Complete the graph using information from scenario descriptions associated to goals having the same parameters as G except the manner,
Step3: Compute all possible missing paths,
Step4: Submit missing paths to RCA. RCA selects the ones of interest and associates each of them with a specific manner to fulfill G,

Comment:

The guiding rule A2 aims to discover alternative goals of a goal G. These discovered goals have the same verb, the same parameters (i.e. source, target, beneficiary, and means) but different manners. *'Withdraw cash from ATM in a normal* *way,' 'Withdraw cash from ATM in a normal way with code error correction'* are examples of goals that rule A2 aims at discovering. This rule allows a graph to be constructed such that it represents all possible paths of actions already identified in the scenario of the initial requirement chunk (step 1) and completed by scenarios of its ORed requirement chunks (step 2) having the same purpose (the verb and parameters are the same except the manner parameter). A path is characterised by *zero* to *n* nested flow conditions and a graph of paths is considered incomplete when there exists a flow condition without a path to look after its negation. For example, in the scenario associated to the goal *'Withdraw cash from the ATM in a normal way'* (see Fig. 5) the path is composed of the four nested following conditions:

```
1 If the card is valid
    2 If the code is valid
        3 If the amount is valid
            4. If the user asked the ATM to supply a receipt
```

Therefore, four missing paths are identified one for each condition. Each of them leads to a case where the condition can be false. A formalization of this algorithm is given in Appendix A. The rule computes all the combinations of negated conditions that should be investigated as possible missing paths (step 3). Each path identifies an alternative manner to deal with the initial goal G. There are two types of scenarios: the normal one and the exceptional one. For each case the RCA is asked to determine whether it is a normal or exceptional one and to propose the manner (step 4). Each discovered goal shall be further concretized by a scenario to complete the requirement chunks. This will be done in the authoring activity.

Composition guiding rule (C1)

Goal:

Discover (from requirement chunk <G, Sc>)$_{So}$
(goals ANDed to G)$_{Res}$
(reasoning on final and initial states of Sc)$_{Ma}$

Body:

Step1: Check inclusion / exclusion of initial states in final states of Sc,
Step2: For every initial state *Is* that is not included in the final states, RCA is asked to point out the final state *Fs* hindering the reaching of the initial state,
Step3: Suggest to the RCA a recovery scenario having *Fs* as part of its initial states and *Is* as part of its final states. RCA is requested to name the associated recovery goal.

Comment:

The guiding rule C1 uses the requirement chunk as a source for reasoning *(from requirement chunk <G, Sc>)$_{So}$* and discovers complementary goals of G *(goals ANDed to G)$_{Res}$* based on the inclusion of final and initial states of the initial scenario *(reasoning on final and initial states of Sc)$_{Man}$*. The body of the rule uses the inclusion property according to which the initial states of a scenario must in its final states for ensuring a self-contained functioning i.e every scenario execution leaves the involved agents in a state which permits the repeated execution of the same scenario. The rule checks if the inclusion property holds (step 1). If this is not the case, then this means that an exceptional state has been reached (step 2). Once the rule has detected the needed recovery scenarios (step 3), it asks the RCA to qualify them by goals to be ANDed to goal G.

Composition guiding rule (C2)

Goal:

Discover (from requirement chunk <G, Sc>)$_{So}$
(goals ANDed to G)$_{Res}$
(reasoning on Sc interactions)$_{Man}$

Body:

Step1: RCA identifies the interaction objects in Sc that correspond to resources,
Step2: Construct interaction pairs (*Consume*, *Produce*) for each identified resource.
Step3: Suggest a new goal ANDed to G for every incomplete pair (i.e. in which either the Consume interaction or the Produce interaction is missing).
Step4: RCA selects the relevant goals and names them.

Comment:

The guiding rule C2 uses a classification of scenario interactions to support the discovery of (*goals ANDed to G*)$_{Res}$, which are complementary to the initial goal G. It identifies interactions which objects are resources (step 1), for example, the *'card'* and the *'cash'*. Applying the producing / consuming principle, for each resource, the rule searches for pairs of interactions in which the one consumes the resource that the other produces (step 2). Every incomplete pair originates a new goal (step 3) being accepted or not by the RCA (step 4).

Refinement guiding rule (R1)

Goal:

Discover (from requirement chunk <G, Sc>)$_{So}$
(goals refined from G)$_{Res}$
(using every interaction of Sc as a goal)$_{Man}$

Body:

Step1: Associate a goal Gi with every atomic action Ai in Sc. Gi refines G.
Step2: Complement Gi by the manner 'in a normal way'.
Step3: RCA evaluates the proposed menu of goals Gi and selects the goals of interest.
Step4: Requirement chunks corresponding to these selected goals are ANDed one another.

Comment:

The guiding rule R1 aims at refining a given requirement chunk *(from requirement chunk <G, Sc>)$_{So}$* by suggesting new goals at a lower level of abstraction than G *(goals refined from G)$_{Res}$*. The refinement mechanism underlying the rule treats every interaction between two agents in the scenario Sc as a goal for the next lower level of abstraction (step 1). For the goals accepted by the RCA (step 3), the corresponding requirement chunks are ANDed to one another (step 4).

Refinement guiding rule (R2)

Goal:

Discover (from requirement chunk <G, Sc>)$_{So}$
(goals refined from G)$_{Res}$
(by completing with actions)$_{Man}$

Body:

Step1: RCA types actions according to classification information provision / request, service provision / request, condition evaluation action / constrained flow of actions).
Step2: Construct action pairs for each of the three types.
Step3: Detect missing actions and update Sc accordingly.
Step4: Suggest Refined goals to G for every added action.
Step5: RCA evaluates and names the selected goals.

Comment:

The guiding rule R2 aims at refining a given *requirement chunk <G, Sc>* by suggesting new actions (*by completing with actions*)$_{Man}$, that could be looked upon as goals refining G: (*goals refined from G*)$_{res}$. The rule uses three classes of dependent action pairs: (service request, service provision), (information request, information provision) and (condition evaluation action, constrained flow of actions). The RCA uses these classes to type the actions of the scenario. There exists a dependency among the two actions of a pair: any service request implies at least one service provision, an information request implies at least one information provision and a constrained flow of actions implies an action which evaluates the condition. Using these dependencies, pairs of actions are constructed (step 2).Every incomplete pair suggests a missing action (step 3) which is inserted in the scenario. Every new action is suggested as a goal for the next step of refinement (step 4). The RCA evaluates and names the selected goals.

4 Applying Rules to the *Crews-L'Ecritoire* Goal Oriented Approach

This section illustrates the use of the generic rules within the *Crews-L'Ecritoire* goal oriented approach. In [29], we have classified scenarios into *contextual*, *system interaction* and *system internal* scenarios. Here, we extend this classification to requirement chunks. The three types of RCs, namely *contextual*, *system interaction* and *system internal requirement chunks* inherit their types from the scenario types and identify three types of associated goals. As a result we organize the requirements collection in a three levels abstraction hierarchy. We believe that this helps separating concerns in requirements elicitation. This was proved useful when applying the approach to the ELEKTRA real case [21]. The following is a walk through these three levels to illustrate the discovery process and the use of the discovery rules with the ATM example. The walkthrough is presented as a use session of *L'Ecritoire* and illustrated with screen dumps.

4.1 Illustrating Goal Discovery at the Contextual Level

The aim of the *contextual level* is to identify the services that a system should provide to an organisation and their rationale. At this level, several alternative architectures of services are postulated and evaluated. All of them correspond to a given business goal. Let *'Improve services to our bank customers'* be such a business goal.

A *contextual chunk* captures a *design alternative* defined by a *design goal* and a *service scenario.* A *design goal* expresses one possible manner of fulfilling the business goal. For example, the design goal *'Provide cash to our bank customers from ATM'* is one possible way of satisfying the business goal. A *service scenario* describes the flow of services among agents (one being the system itself) which are felt necessary to fulfill the design goal. An atomic action of a service scenario is a service such as 'the bank customer withdraws cash from the ATM' whereas the entire scenario describes the services architecture associated with the design goal. The requirement chunk RC1 in Fig. 5 is an example of contextual chunk.

At the contextual level, it is of major importance to explore as many design alternatives as possible i.e., to visualize the various alternative ways by which a system can help an organization to achieve one of its objectives. The guiding rule A1 is useful for this purpose. The screen dump in Fig. 8 presents the window provided by *L'Ecritoire* to support the application of this rule to RC1.

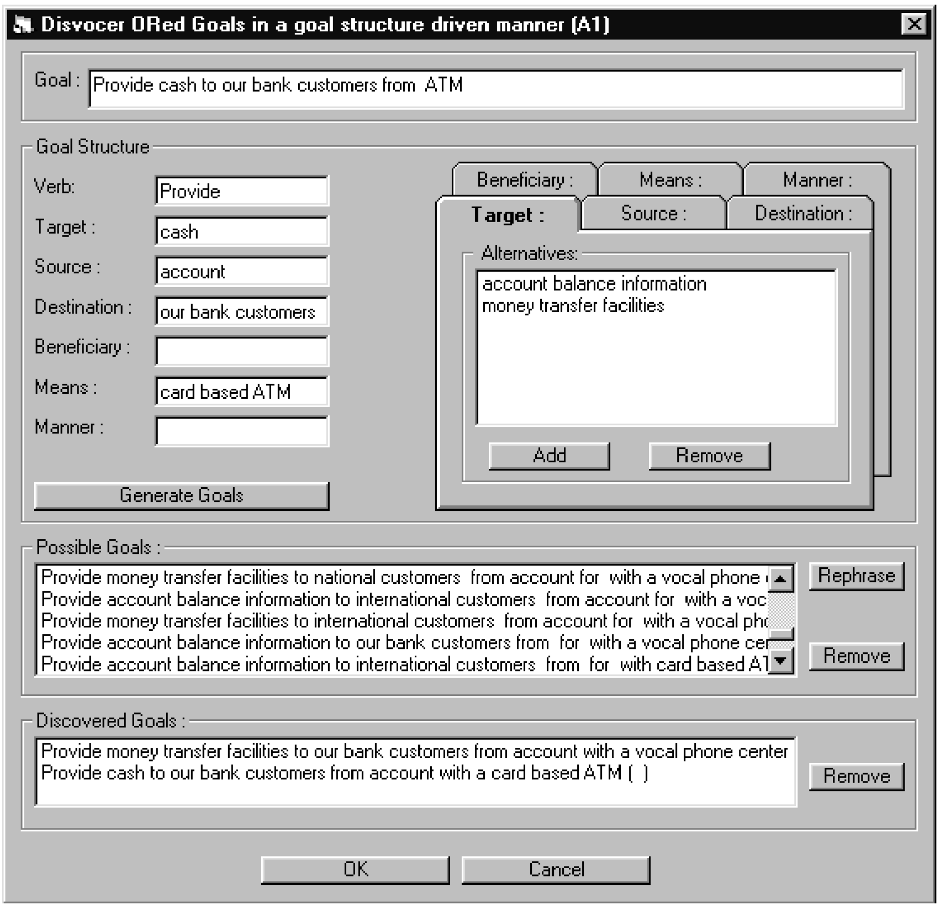
First, the rule helps the RCA to identify the design goal parameters (see 'Goal Structure' frame). Secondly, the RCA is asked to provide alternative values for each parameter. As shown in the right top part of the window, the RCA has identified 'account balance information' and 'money transfer facilities' as two possible alternative target values.

When the 'Generate Goals' button is used, all possible combinations of values of parameters are computed, combined with the verb 'Provide' and presented to the RCA parameter value wise (see the frame 'Possible Goals' in Fig. 8). There exists a predefined order of parameter types that can be customized in every instantiation of the rule. Assuming the following order for the current example: <target, beneficiary, source, means>, the possible combinations will be presented in this order; i.e., for a given target, all possible beneficiaries, for a given 2-tuple <target, beneficiary> all possible sources and for a given 3-tuple <target, beneficiary, source>, all possible means.

This leads to 27 alternative manners to fulfill the business goal *'Improve services to our bank customers'*. These manners are themselves[3] ORed to the RC1's goal and stored in the RCs hierarchy (Fig. 9). The goal selected by the RCA are displayed in the bottom frame of the window. The payoff of applying rule A1 is a semiautomated generation of alternative design options. The rule is an incentive to envision various design solutions and explicitly choose one, then avoiding implicit choices which can demonstrate to be wrong later on. The parameter value wise strategy to generate goals was found useful by practitioners as it limits the combinatorial space of solutions.

Following the suggestion of *L'Ecritoire*, the RCA writes the service scenario Sc1 as shown in Fig. 5 and decides to explore in more detail the requirements of the system characterized by RC1. This is achieved by moving to the interaction level.

3. For sake of readability we reduce the RCk^i goal statements to the expression of manners. The complete expression, for example for RC1, is *'Improve (services)$_{Obj}$ (to our bank customers)$_{Ben}$ (by providing (cash)$_{Obj}$ (to our bank customers)$_{Dest}$ (from ATM)$_{Mea}$)$_{Man}$'*. It uses a recursive definition of the goal manner. The manner *'by providing cash to our bank customers from ATM'* associated to 'Improve' is itself a goal expression with an object, the 'cash', a destination 'our bank customer' and a means, the 'ATM'.



Goal : Provide cash to our bank customers from ATM

Goal Structure

Verb:	Provide
Target :	cash
Source :	account
Destination :	our bank customers
Beneficiary :	
Means :	card based ATM
Manner :	

Alternatives:
account balance information
money transfer facilities

Possible Goals :
Provide money transfer facilities to national customers from account for with a vocal phone
Provide account balance information to international customers from account for with a voc
Provide money transfer facilities to international customers from account for with a vocal ph
Provide account balance information to our bank customers from for with a vocal phone ce
Provide account balance information to international customers from for with card based A

Discovered Goals :
Provide money transfer facilities to our bank customers from account with a vocal phone center
Provide cash to our bank customers from account with a card based ATM ( )

Fig. 8. Example of application of rule A1 using *L'Ecritoire*.

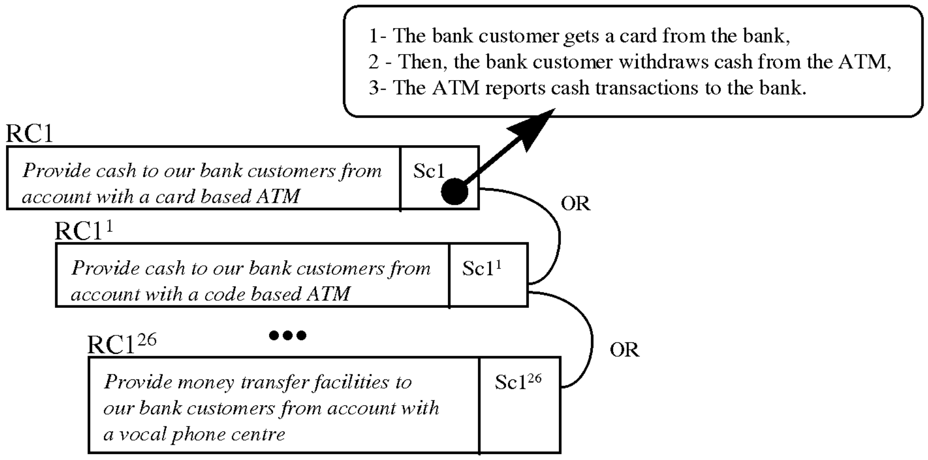

1- The bank customer gets a card from the bank,
2 - Then, the bank customer withdraws cash from the ATM,
3- The ATM reports cash transactions to the bank.
RC1
Provide cash to our bank customers from account with a card based ATM
Sc1
OR
RC1¹
Provide cash to our bank customers from account with a code based ATM
Sc1¹
•••
RC1²⁶
Provide money transfer facilities to our bank customers from account with a vocal phone centre
Sc1²⁶
OR

Fig. 9. Contextual chunks discovered from the contextual chunk RC1.

4.2 Illustrating Goal Discovery at the System Interaction Level

At the *system interaction level* the focus is on the interactions between the system and its users. These interactions are required to achieve the services assigned to the system at the contextual level. Each of these services are refined in system interaction chunks and new ones are added.

A *system interaction chunk*, captures one way of providing a service as expressed at the previous level. It couples a *service goal* and a *system interaction scenario*. A *service goal* expresses a manner of providing a service, for example '*Withdraw cash from ATM in a normal way*' and therefore, establishes a refinement link with a contextual chunk. The associated *system interaction scenario* describes a flow of interactions between the system and its users to fulfill the service goal. Sc1.1 in Fig. 5 is an interaction scenario which, coupled to the service goal G1.1, constitutes the requirement chunk RC1.1.

The guiding rule R1 helps the RCA in the discovery of service goals by analyzing the service scenario Sc1. First, every Sc1 action is proposed as a candidate service goal. Second, those which are selected by the RCA are rephrased as goals. For example, action number 2 in Sc1 (see Fig. 9) is rephrased as '*Withdraw cash from ATM*'. Third, the rule suggests that the scenarios authored for these goals describe the normal courses of actions. Thus, the manner of every generated goal is fixed to *'in a normal way'*. This leads in our example to the three refined goals: G1.1, G1.2, and G1.3. The corresponding RCs (called RC1.1, RC1.2, and RC1.3 in Fig. 12) are ANDed to one another and related to RC1 through a refinement link.

For each of the three service goals, scenarios are authored and their contents help discovering new goals which, when associated with scenarios, support the discovery of new goals etc. This is made possible by the rules C1, C2, and A2. The first two of these guide the RCA in the discovery of goals ANDed to the three RCs whereas A2 guides the discovery of ORed RCs. Referring to the use case terminology, one can say that the rule A2 helps identifying the variations of a normal course of actions in a use case whereas rules C1 and C2 help discovering the use cases complementary to the one under description. These use cases are necessary to obtain a complete description of the system functionalities.

As illustrated in Fig. 10, the rule A2 computes and displays all possible missing paths of actions in the scenario of RC1.1. The assumption is that each of these paths can be regarded as a different way of fulfilling the goal G1.1. For example, the path 'C1 and not C2' in Fig. 10 is associated to the manner *'in a normal way with code error correction'* attached to the goal '*Withdraw cash from ATM*'.

The application of A2 in our example leads to the introduction of $RC1.1^1$, $RC1.1^2$, $RC1.1^3$, and $RC1.1^4$ which are ORed to RC1.1 as shown in Fig. 12.

Once the scenarios corresponding to these new goals have been authored, the rule A2 can be applied again to discover new manners of withdrawing cash. For example, the goal $G1.1^8$, *'Withdraw cash from ATM by treating the exception of three invalid code attempts'* is discovered (Fig. 10).

The rule C1 is then applicable to the scenario associated with this last goal. The initial and final states in $Sc1.1^8$ are the following:

- *Initial State*: The ATM is ready. The user has a card.
- *Final State*: The ATM is ready. The ATM has the user's card.

The inclusion property does not hold. Indeed an exceptional state has been reached (the ATM has the user's card). Thus, a recovery scenario is needed. In our example the rule suggests a goal for the restoration of the card to the user. The RCA names the goal *'Restore card to user'*. The goal is associated within RC1.6 (Fig. 12) to a scenario sketching the way the card can be restored. This scenario has 'The ATM has the user's card' in its initial states and 'The user has a card' in its final states.

Finally, let us apply the rule C2 to RC1.1. The rule uses the resource consumer/producer principle in order to detect two ANDed goals to G1.1.

As illustrated in Fig. 11, the rule displays the actions of the scenario in the 'Actions List' and asks the RCA to separate those which consume resources ('Consuming' frame) from those which produce resources ('Producing' frame). For example, actions 8 and 9 are productions because the ATM produces a resource when it 'delivers the cash to the user' or when 'a receipt is printed'. Then, the rule asks the RCA to identify producing/consuming pairs (displayed in the 'Producing/Consuming Completed Pairs' frame). Actions that cannot participate in complete pairs remain in the 'Producing' and 'Consuming' frames.

In Fig. 11 there are two remaining actions in the 'Producing' frame. Therefore, two new goals are suggested, namely:

- 'Fill in the ATM with receipt paper' (G1.4), and
- 'Fill in the ATM with cash' (G1.5).

The corresponding requirement chunks RC1.4 and RC1.5 are created and ANDed to RC1.1 (Fig. 12).

The set of requirement chunks resulting of the application of rules C1, C2, and A2 at the system interaction level, is summed up in Fig. 12. Clearly, the refinement process at this level was initiated from the scenario Sc1, by the application of rule R1. However, it shall be noticed that composition and alternative rules were useful to reach a more complete collection of system requirements. Still, more detailed requirements have to be found and this is supported by the application of the generic rules at the system internal level.

4.3 Illustrating Goal Discovery at the System Internal Level

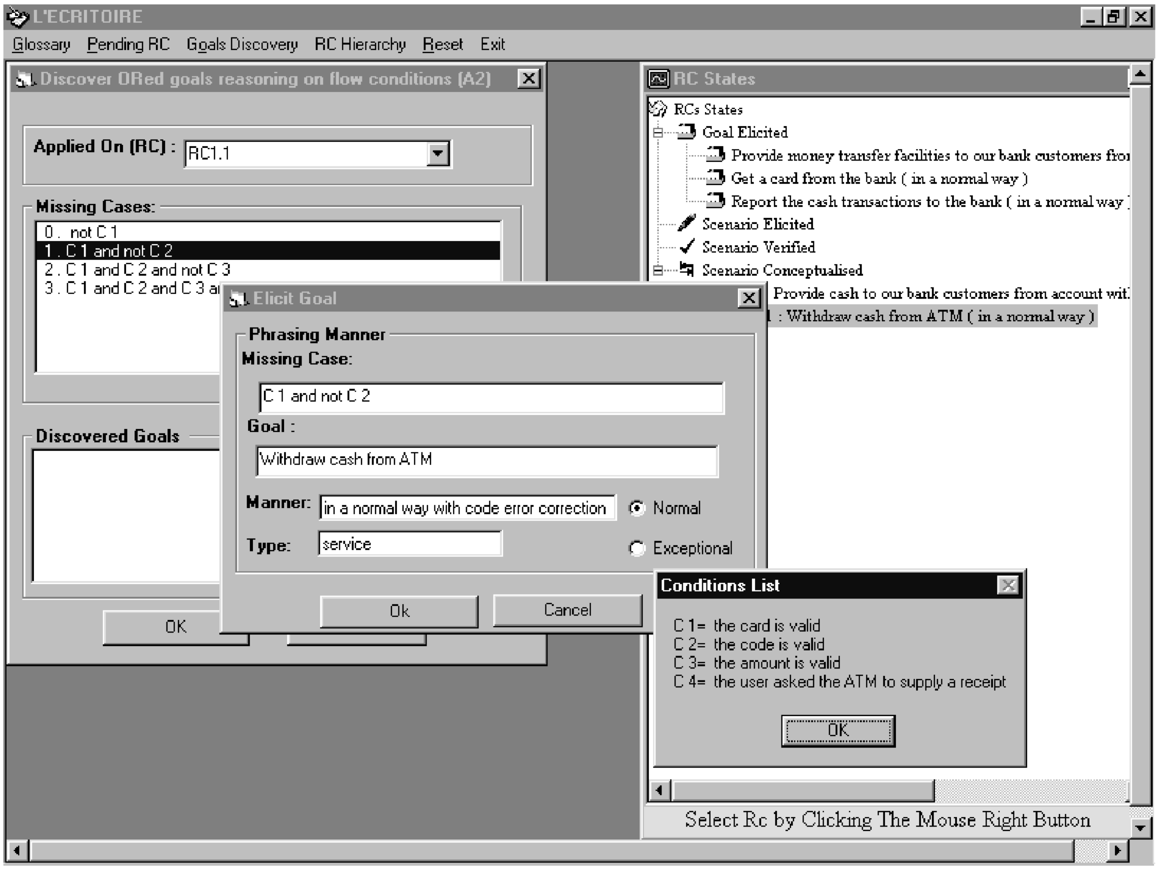
The *system internal level* focuses on what the system needs, to perform the interactions selected at the system interaction level. The *'what'* is expressed in terms of internal system actions that involve system objects but may require external objects such as other systems. System interactions are refined in internal system chunks and new ones are added.

An *system internal chunk* combines a *system goal* and a *system internal scenario*. A *system goal* expresses a possible manner to perform an action identified in a system interaction scenario. For example, *'Verify the card validity in a normal way'* is a system goal. The associated *system internal scenario* describes the flow of interactions among the system objects to fulfill the system goal (see Fig. 13). The couple constitutes the requirement chunk RC1.1.1.

Applying the rule R1 to the requirement chunk RC1.1 results in chunks RC1.1.1 to RC1.1.4. Fig. 13 shows the result of the discovery activity: alternative and complementary system internal RCs are discovered by applying the rules R2 (to discover RC1.1.5 and RC1.1.6), A2 (to discover $RC1.1.1^1$ to $RC1.1.1^3$), C2 (to discover RC1.1.7) and C1 (to discover RC1.1.8). For the sake of space we do not describe this process in detail. However, the result can be retrieved by applying the mentioned rules to the scenario Sc1.1.1.

5 The Potential of the Approach

This section evaluates the potential practical benefits that one can expect from the use of the *Crews-L'Ecritoire* approach. In order to relate these benefits to industrial practice, three workshops were conducted with participation from system development experts drawn from French industry. These experts where asked to evaluate the approach. Thus, the potential of the approach as presented here has been done by uninvolved parties.



Discover ORed goals reasoning on flow conditions (A2)

Applied On (RC) : RC1.1

Missing Cases:

0 . not C 1
1 . C 1 and not C 2
2 . C 1 and C 2 and not C 3
3 . C 1 and C 2 and C 3 a

Discovered Goals

RC States

RCs States
Goal Elicited
Provide money transfer facilities to our bank customers fro
Get a card from the bank (in a normal way)
Report the cash transactions to the bank (in a normal way
Scenario Elicited
Scenario Verified
Scenario Conceptualised
Provide cash to our bank customers from account wi
1 : Withdraw cash from ATM (in a normal way)

Elicit Goal

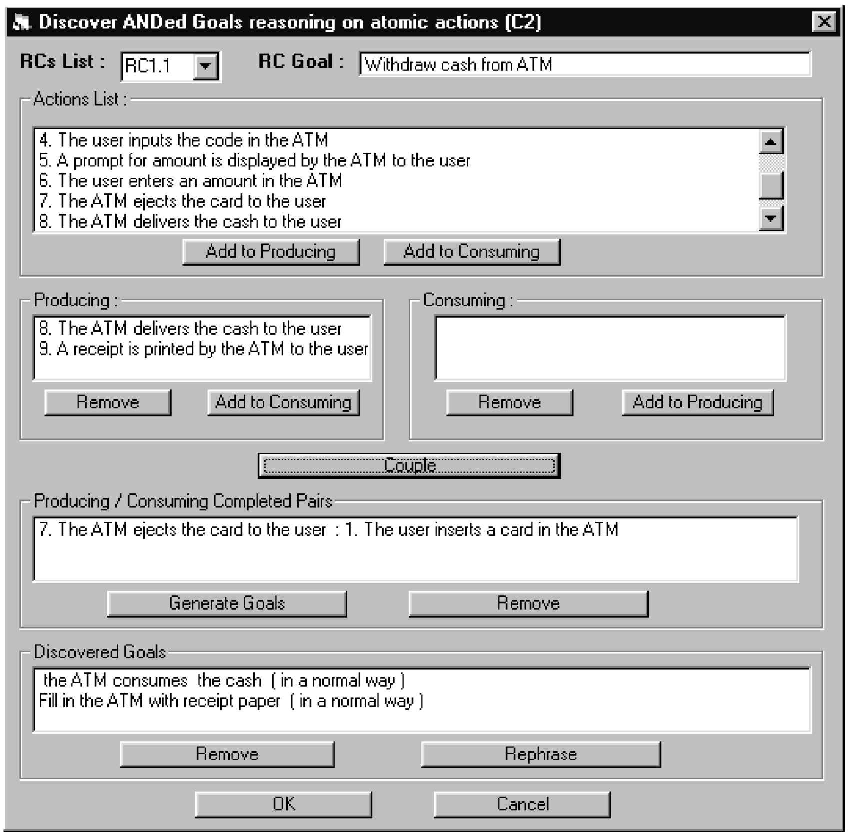
Phrasing Manner
Missing Case: C 1 and not C 2
Goal : Withdraw cash from ATM
Manner: in a normal way with code error correction [✓ Normal] Exceptional
Type: service

Conditions List

C 1= the card is valid
C 2= the code is valid
C 3= the amount is valid
C 4= the user asked the ATM to supply a receipt

Select Rc by Clicking The Mouse Right Button

Fig. 10. Example of application of rule A2 using *L'Ecritoire*.



Discover ANDed Goals reasoning on atomic actions (C2)

RCs List : RC1.1 RC Goal : Withdraw cash from ATM

Actions List :

4. The user inputs the code in the ATM
5. A prompt for amount is displayed by the ATM to the user
6. The user enters an amount in the ATM
7. The ATM ejects the card to the user
8. The ATM delivers the cash to the user

Producing :

8. The ATM delivers the cash to the user
9. A receipt is printed by the ATM to the user

Consuming :

Producing / Consuming Completed Pairs

7. The ATM ejects the card to the user : 1. The user inserts a card in the ATM

Discovered Goals

the ATM consumes the cash (in a normal way)
Fill in the ATM with receipt paper (in a normal way)

Fig. 11. Example of application of rule C2 using *L'Ecritoire*.

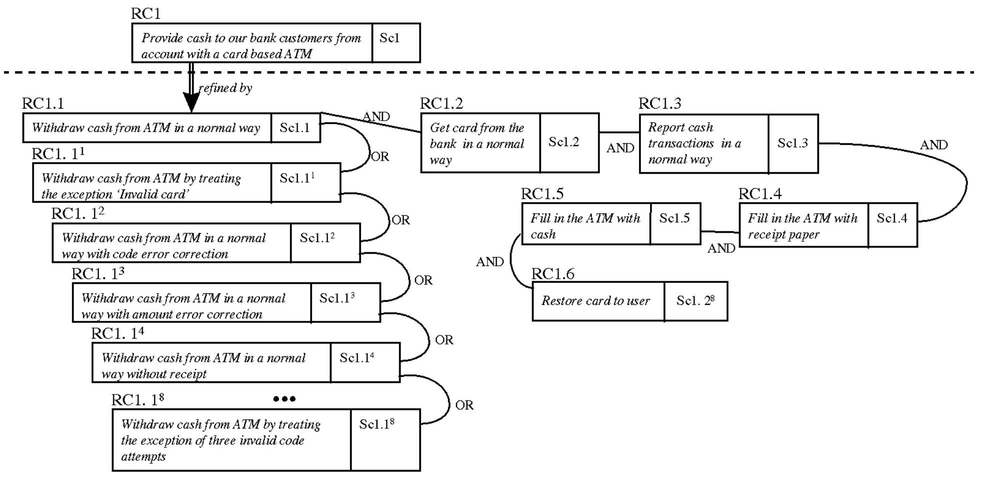
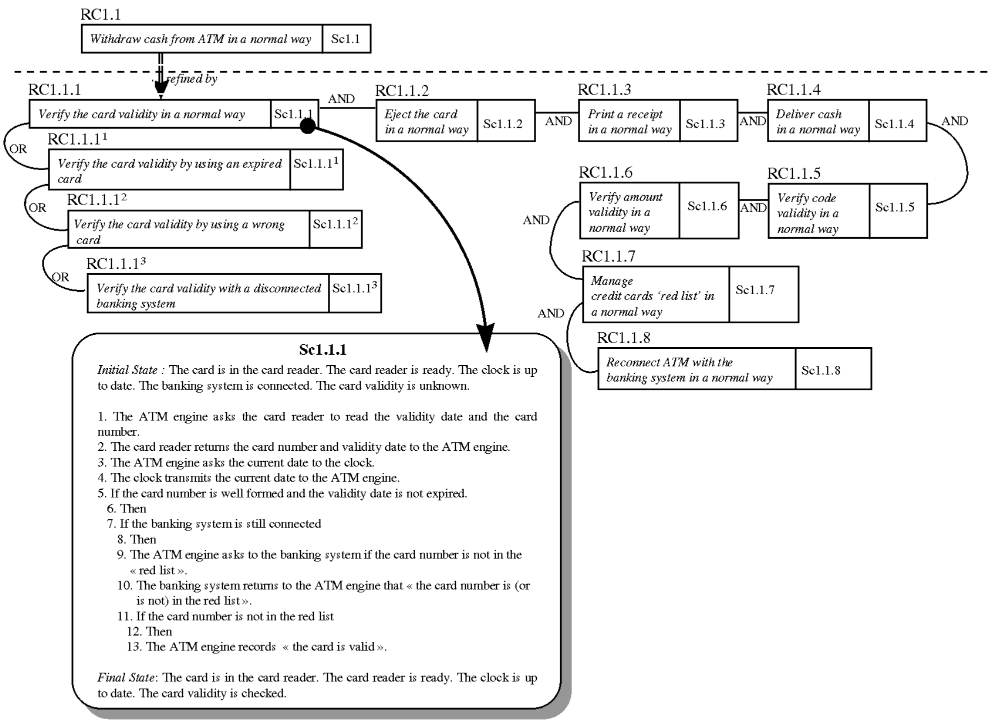

RC1
Provide cash to our bank customers from account with a card based ATM
Sc1
refined by
RC1.1
Withdraw cash from ATM in a normal way
Sc1.1
RC1. 1¹
Withdraw cash from ATM by treating the exception 'Invalid card'
Sc1.1¹
RC1. 1²
Withdraw cash from ATM in a normal way with code error correction
Sc1.1²
RC1. 1³
Withdraw cash from ATM in a normal way with amount error correction
Sc1.1³
RC1. 1⁴
Withdraw cash from ATM in a normal way without receipt
Sc1.1⁴
RC1. 1⁸
Withdraw cash from ATM by treating the exception of three invalid code attempts
Sc1.1⁸
OR
AND
RC1.2
Get card from the bank in a normal way
Sc1.2
RC1.3
Report cash transactions in a normal way
Sc1.3
RC1.5
Fill in the ATM with cash
Sc1.5
RC1.4
Fill in the ATM with receipt paper
Sc1.4
RC1.6
Restore card to user
Sc1. 2⁸

Fig.12. System interaction chunks discovered from the contextual chunk RC1.


RC1.1
Withdraw cash from ATM in a normal way
Sc1.1
refined by
RC1.1.1
Verify the card validity in a normal way
Sc1.1.1
RC1.1.1¹
Verify the card validity by using an expired card
Sc1.1.1¹
RC1.1.1²
Verify the card validity by using a wrong card
Sc1.1.1²
RC1.1.1³
Verify the card validity with a disconnected banking system
Sc1.1.1³
OR
AND
RC1.1.2
Eject the card in a normal way
Sc1.1.2
RC1.1.3
Print a receipt in a normal way
Sc1.1.3
RC1.1.4
Deliver cash in a normal way
Sc1.1.4
RC1.1.6
Verify amount validity in a normal way
Sc1.1.6
RC1.1.5
Verify code validity in a normal way
Sc1.1.5
RC1.1.7
Manage credit cards 'red list' in a normal way
Sc1.1.7
RC1.1.8
Reconnect ATM with the banking system in a normal way
Sc1.1.8

Sc1.1.1

Initial State : The card is in the card reader. The card reader is ready. The clock is up to date. The banking system is connected. The card validity is unknown.

1. The ATM engine asks the card reader to read the validity date and the card number.
2. The card reader returns the card number and validity date to the ATM engine.
3. The ATM engine asks the current date to the clock.
4. The clock transmits the current date to the ATM engine.
5. If the card number is well formed and the validity date is not expired.
   6. Then
   7. If the banking system is still connected
      8. Then
      9. The ATM engine asks to the banking system if the card number is not in the « red list ».
      10. The banking system returns to the ATM engine that « the card number is (or is not) in the red list ».
      11. If the card number is not in the red list
         12. Then
         13. The ATM engine records « the card is valid ».

Final State: The card is in the card reader. The card reader is ready. The clock is up to date. The card validity is checked.

Fig. 13. System internal chunks discovered from the system interaction chunk RC1.1.

Before conducting these workshops, an exploratory survey of practice was conducted by the CREWS consortium through site visits. This survey looked at 15 projects in four European countries. The results of the survey have been reported in [33]. This survey highlighted three major problems and the purpose of the one day workshops was to evaluate whether or not the *Crews-L'Ecritoire* approach alleviates any of these problems. This section is organized in two parts, the first of which deals with the three major problems and how the *Crews-L'Ecritoire* approach tries to solve these. The second part presents the feedback received in the one-day workshops.

5.1 Meeting Industrial Requirements

The three major problems identified were as follows:

1) *methodological support*. As reported in [33], practitioners feel a very high need for methodological guidance. In the *Crews-L'Ecritoire* approach, guidance is provided by using the relationship (G, Sc) in the forward direction to operationalize goals and in the reverse direction, to discover goals. Guidance to discover goals is flexible as it provides three complementary and independent strategies:

- using the refinement strategy, goals lower than G in the goal hierarchy are discovered. This is particularly useful in those situations in which the scenario associated with a goal contains actions that cannot be directly operationalized. In such a case, a statement needs to be made about how these high level actions can themselves be made operational. The approach proposes to treat all such actions as goals. As a consequence, scenarios can be associated with these goals and the associated scenarios can be further analysed to determine if their actions can be directly made operational or not. This goal discovery—scenario formation—goal discovery cycle continues till all actions of scenarios can be made directly operational.

- using the AND / OR strategies, goals which are alternative of or complementary to G are discovered. This is useful in achieving completeness, i.e., in determining the complete set of scenarios in a use case and the set of use cases necessary to cover all the features of the system under investigation. The guidance provided to discover goals is flexible in the sense that there is no predefined, imposed order in which the process has to be performed but instead a dynamic selection of one among the three strategies can be done by the RCA at each step.
- the automation of the guiding rules overcomes the heavy investment that their manual application would require.

2) It addresses [33] *top-down decomposition of scenarios*, 'from *informal to formal* scenario' definitions, and 'from *black-box to white-box* scenario' development.

Observation of current practice shows that the majority of requirements stakeholders prefer to develop scenarios in a *top down manner*. This is also true in goals modeling [7], [13]. The difficulty is in the control of the top down decomposition. The *Crews-L'Ecritoire* approach is about the discovery of subgoals from actions of the scenario attached to the goal. The process is therefore top-down but controlled by the refinement guiding rules. We have seen in Section 4 of the paper how the refinement relationship helps in moving from contextual RCs dealing with contextual issues such as alternative design goals and related services to more detailed issues like service goals and system interactions descriptions.

Again, in practice, stakeholders like to apply the *black box/white box principle*. However, when put in practice, the difficulty with this principle is to ensure that the level of abstraction of the black box is preserved when its insides are being described. For example, it was reported that in many projects [33], authors mix up different levels of detail and different concerns in the same scenario description. This risk is removed in our approach since the abstraction level of a scenario is determined by its goal and is made explicit by goal discovery, prior to scenario authoring. Besides, the latter is guided by content rules which help in understanding the nature of the information required in the scenario.

Clearly, the approach contributes to the mapping from *informal to formal scenario descriptions*. Indeed we assume scenarios to be textual and use authoring rules [3], [28] to provide style and contents guidelines as well as devices for analysis, disambiguation, and completion. The linguistic devices are based on a case grammar and linguistic patterns. Thus, they support the transformation of an informal scenario description written in full prose into an unambiguous, complete, and well structured text.

3) The *Crews-L'Ecritoire* approach addresses all the three problems found by Cockburn [5] in trying to structure use cases with goals in relatively large projects. These are as follows:

- *difficulty with levels* and the complications encountered in using a goal refinement approach. The *Crews-L'Ecritoire* approach formally tracks goals at different granularity levels in the RCs hierarchy. This is achieved by the Refinement relationship. The value addition done by our approach lies in goal discovery rules which exploit the reverse relationship between G and Sc, i.e., from scenario to goals, in order to discover goals lower in the RCs hierarchy. Rules of the refinement strategy help in systematising the operationalization of goals as an independent activity, separate from that of relating goals through AND / OR relationships.
- the unsatisfactory and 'ad hoc' process of identifying variations of a use case. Cockburn notices that "each subordinate goal case carries forward the list of variations, until it is time to break them out into their own use cases," In the *Crews-L'Ecritoire* approach variations of the same goal are structured through OR relationships and rules systematize the breaking out of a goal into its variations. Thus, each of these can be concretised by its own scenario.
- the tedious problem of tracking system features across use cases. In practice 'a goal or use case is delivered for a given set of features' and 'the features cross multiple use cases'. Therefore, tracking that all system features are captured in use cases is complicated and the risk of incompleteness of the requirements specification is very high. Our approach solves the problem by attaching features to goals, systematizing the discovery of features and related goals, and by defining guiding rules leading to an AND goal structure. The simplest feature is developed first and others are discovered using the AND goal discovery strategy.

5.2 Industrial Evaluation of the Approach

The workshops held to evaluate the *Crews-L'Ecritoire* approach include a presentation of the approach, a demonstration of the tool, a session of tool manipulation by the participants and close with a panel discussion. At the end of the workshops, participants were asked to fill-in a questionnaire. The results extracted from these questionnaires are summed up in Tables 1 and 2.

Participants were asked to grade the usefulness of the *Crews-L'Ecritoire* approach on a scale of 1 to 7. The overall grading is presented in Table 1. It shows a quite high level of satisfaction, which is encouraging.

Table 2 shows the four more frequently mentioned contributions of the approach to the improvement of the current practice. There is clearly convergence with the industrial problems reported in 5.1, especially for points (A) and (D).

The *methodological support* was most frequently mentioned benefit expected from the use of the approach. More precisely:

TABLE 1
AVERAGE GRADING OF USEFULNESS

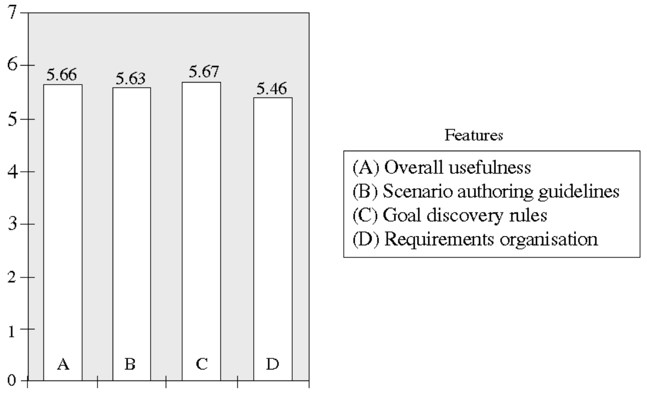
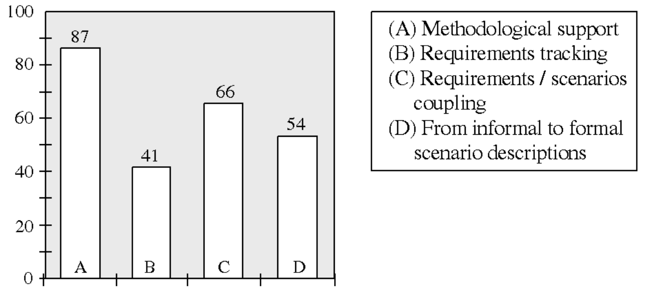


TABLE 2
FREQUENCY OF IDENTIFIED POTENTIAL BENEFITS



- the AND / OR strategies were found useful in achieving completeness of system functional requirements and system physical requirements,
- the OR strategy helped in the identification of variations of normal system behaviors,
- rule A1 was found useful in cases where alternative designs have to be envisioned,
- over all, the top-down approach of the RE process was appreciated as fitting the natural practice. The difficulty in controlling the top down decomposition process in practice was found mitigated by the guiding rules.

Participants noticed that the methods they know about such as OOSE [16] and SOMATiK [12] are lacking a *tight coupling between requirements and scenarios* which is provided in the *Crews-L'Ecritoire*.

Positive statements were made regarding the contribution of the approach to the mapping *from informal to formal scenario descriptions*. The linguistic devices for an analysis, disambiguation and completion were found applicable to most of real situations where it is necessary to support the transformation of an informal scenario description written in full prose into an unambiguous, complete, and well structured text.

Finally, the predefinition of the three levels contextual, system interaction, system internal was found useful in clarifying the concerns of system requirements. The principle of tracking system requirements from business goals, and alternative designs to system functional and physical requirements was appreciated, but the technical support for this was found limited.

Extensions and improvements such as supporting several natural languages, connection with industrial tools (i.e. RequisitePro, Doors), providing cooperative negotiation and evaluation of requirements were suggested by workshop participants.

6 CONCLUSIONS

The basis of our approach is the exploitation of the goal-scenario relationship but in the reverse direction. One can now talk of a tight coupling between goals and scenarios; in the forward direction this coupling promotes goal operationalization whereas in the reverse direction it promotes goal discovery. Since, in the forward direction, scenarios represent a concrete, useful way of realizing a goal, any technique which uses scenarios to discover goals shall produce only useful goals. This contributes to removing the fitness of use problem identified by Potts [25] which leads to the generation of spurious, uninteresting, or noncritical goals.

Since interactions expressed in scenarios are concrete and recognizable, the use of goal-scenario coupling for goal discovery helps in removing the 'fuzziness' that domain experts find in the notion of a goal. Instead, each interaction corresponds to goals. Again, goal discovery now becomes a natural process through interactions of scenarios and the goal-scenario coupling removes some of the mystery and ad hocism associated with it. In this sense it helps in goal discovery.

Finally, the combination of composition, alternative and refinement rules alleviates the problem of goal reduction. By generating AND, OR relationships and goals at different abstraction levels, the approach automates and supports a major part of the requirements engineer's work.

APPENDIX A

We present here: 1) formal definitions of elements of the requirement chunk model, and 2) formal definitions of some steps of the guiding rules.

DEFINITION 1 (Requirement Chunk). *A* requirement chunk RC *is a couple* <G, Sc> *where* G *and* Sc *are, respectively, a goal and a scenario*[4] *as defined in Definitions 2 and 3. We use in the following, the functions goal and scenario: goal:*

RC → G,
scenario: RC → Sc,

such that RC *is the set of requirement chunks,* G *the set of goals,* Sc *the set of scenarios, and goal(*RCi*) (respectively scenario*(RCi)*) gives the goal* Gi *(the scenario* Sci*) associated to* RCi.

Moreover, the functions AND, OR, and RefinedBy:

AND: RC → RC
OR: RC → RC
RefinedBy: RC → RC,

give the RCs associated to a given RC, respectively, through AND, OR and RefinedBy relationships. The AND, OR, and RefinedBy functions are bijective.

4. RC, G, and Sc are, respectively, the sets of the requirement chunks, goals, and scenarios.

In the following, we make use of AND^{trans} and OR^{trans} which, given a requirement chunk RCi, give all the requirement chunks that are complementary (respectively alternative) to RCi by transitive application of the AND function (respectively of the OR function), and all the requirement chunks to which RCi is ANDed (respectively ORed) by transitive application of AND^{-1} (of OR^{-1}).

DEFINITION 2 (Goal). *A* goal g *is an element of the set* $G / (\exists\ RCi \in RC / goal(RCi) = g) \vee (\exists\ g' \in G / manner(g') = g)$.

Indeed, a goal is either coming from a requirement chunk, or it is embedded in the manner parameter of a goal.

Moreover, g is such that there exists a verb $v \in VE$ and a target $t \in TA / (verb(g) = v) \wedge (target(g) = t)$; verb and target are thus the functions returning the verb and target expressed in the goal g. Indeed, g has, at least, a verb and a target, as informally illustrated in Section 2.1.1.

Additionally, optional parameters can be expressed in goals. These parameters are obtained by the following partial functions:

direction: $RC \rightarrow Dir$,
way: $RC \rightarrow Wa$,
beneficiary: $RC \rightarrow Ben$,
object: $RC \rightarrow Obj$,
result: $RC \rightarrow Res$,
source: $RC \rightarrow So$,
destination: $RC \rightarrow Dest$,
means: $RC \rightarrow Mea$, and
manner: $RC \rightarrow Man$.

DEFINITION 3 (Scenario). *A* scenario *is a graph* $G(X, U, L^x, L^u)$ *where*:

- $X = \{x_1, ..., x_n\}$ *is a finite set of nodes.*
- $U = \{u_1, ..., u_n\}$ *is a finite set of edges where* $u_i \in (X \times X)$.
- L^x *is the node label function* $X \rightarrow Condition$ *where* Condition *is the set of conditions of a scenario:* $Condition \in \{c_i / i \in N\} \cup \{true\}$. L^x *is a total function, and true is used to express unconditioned flows of actions.*
- L^u *is the edge label function* $U \rightarrow Action$, *where* Action *is the set of actions of the scenario:* $Action \in \{a_i / i \in N\} \cup \{Nac\}$. L^u *is a total function, and the null action is used for flow of actions combinations.*

In addition, we define the scenarios initial and final nodes.

- The set X^i of *initial nodes* of a scenario is defined as $\{x_i \in X$ such that: $\neg\exists\ x_j \in X, x_j = f(x_i)\}$, where $f: X \rightarrow X$ is the function following. Indeed there is no node in the scenario graph that precedes any of the initial nodes.
- The set X^f of *final nodes* of a scenario is defined as $\{x_i \in X$ such that: $\neg\exists\ x_j \in X, x_j = p(x_i)\}$, where $p: X \rightarrow X$ is the function preceding. Indeed, there is no node in the scenario graph that follows any of the final nodes.

Within the set of final nodes of a scenario, we distinguish X^{fn} from X^{fe}, where X^{fn} is the set of normal final nodes (which mark a normal end for a scenario, that is to say a scenario end in which the goal associated to the scenario is fulfilled, see Section 2.1.2), and X^{fe} is the set of exception final nodes (which mark an exceptional end for a scenario, that is to say a use case end in which the goal associated to the scenario is not fulfilled, see Section 2.1.2).

The graph G which describes a scenario respects the following constraints:

- G has one and only one initial node: $Card(X^i) = 1$.
- G has one and only one end node: $Card(X^f) = 1$. Thus, the end node of a scenario is either normal or exceptional: $X^{fn} \cup X^{fe} = X^f$.
- There is at least one path from the initial node X^i to the end node X^f.
- G has no loop.

In the remaining, we describe formally some automated steps of the guiding rules presented in Section 3.3.

Guiding rule A1

For the steps 3 and 4, the following formula are applied:

Given an initial goal g, the RCA has provided for each parameter a set of alternative values A_1 to A_7, where A_1 = alternativeObject(g), A_2 = alternativeResult(g), etc. The set of possible combinations is $G^1 = A_1 \times A_2 \times ... \times A_7$.

Once the RCA has applied the selection σ1 to the set of goals (step 5), the hierarchy of requirement chunks is updated such that:

$\forall\ g' \in G^2, G^2 = \sigma1(G^1), \exists\ RC', RC'' \in RC /$
$(g' = goal(RC')) \wedge (g = goal(RC'')) \wedge (RC' \in OR^{trans}(RC''))$.

Guiding rule R1

Steps 1 and 2 are formally described as follows:

Given an initial requirement chunk RC <G, Sc>, the set of generated goals G^1 is such that:

$\forall\ u_i \in U(sc), \forall\ a_i \in L^u(ui), \exists\ g' \in G^1 /$
$(verb(g') = actionName(a_i)) \wedge (target(g') = parameter(a_i))$
$\wedge\ (source(g') = fromAgent(a_i))$
$\wedge\ ((destination(g') = toAgent(a_i))$
$\vee\ (beneficiary(g') = toAgent(a_i))$
$\wedge$ (manner(g') = 'in a normal way').

Once the RCA has applied the selection σ2 to the set of goals (step 4), the hierarchy of RCs is updated such that:

$\forall\ g'' \in G^2, G^2 = \sigma2(G^1), \exists\ RC', RC'' \in RC / (g'' = goal(RC''))$
$\wedge\ ((RC'' \in AND^{trans}(RC')) \wedge (RC' = RefinedBy(RC))$
$\vee\ (RC'' = RefinedBy(RC)))$.

Guiding rule A2

Steps 1 and 2 are formally described as follows:

Given a requirement chunk RC <G, Sc>, let be RCor the set of requirement chunks ORed to RC, that is to say such that:

$\forall\ RC_i \in RCor, (RCi \in OR^{trans}(RC))$
$\wedge\ (manner(goal(RC_i)) = manner(g))$
$\wedge\ (beneficiary(goal(RC_i)) = beneficiary(g))$
$\wedge\ (object(goal(RC_i)) = object(g))$
$\wedge\ (result(goal(RC_i)) = result(g))$
$\wedge\ (source(goal(RC_i)) = source(g))$
$\wedge\ (destination(goal(RC_i)) = destination(g))$
$\wedge\ (means(goal(RC_i)) = means(g))$

The computation of missing paths is done upon a graph gr $(X', U', L^{x'}, L^{u'})$ where $X', U', L^{x'}, L^{u'}$ are defined like X,, U, L^x, L^u respectively, where

- $X' = \cup X_i$, where X_i is the set of nodes associated to scenario(RC_i), $\forall$ RCi $\in$ RCor $\cup$ {RC}
- $U' = (U_i$, where U_i is the set of edges associated to scenario(RC_i) $\forall$ RCi $\in$ Rcor $\cup$ {RC}
- $L^{x'}$ is the node label function $X' \rightarrow C_i$, where C_i is the set of conditions of a scenario scenario(RC_i), $\forall$ $RC_i \in$ RCor $\cup$ {RC}
- $L^{u'}$ is the node label function $U' \rightarrow A_i$, where A_i is the set of actions of a scenario (RC_i), $\forall$ $RC_i \in$ RCor $\cup$ {RC}

This graph does not correspond to the behavior of one scenario, but of several complementary scenarios. Thus, it has the following properties:

- The graph gr has one and only one initial node: Card(X^i) = 1.
- The graph gr has one or several end nodes: Card(X^f) ≥ 1. The end nodes of the graph gr can be normal and exceptional, but there is at least one normal end node: Card (X^{fn}) ≥ 1.
- There is at least one path from the initial node X^i to any end node X^f.
- The graph gr has no loop.

The missing cases sc_i are identified by all paths from X^i to all node in negation(u_i) $\forall u_i \in U'$ / Card(f(u_i)) = 1, where f is the function following, and negation the function returning all possible ways to express $\neg u_i$. Then, each missing case c of sc_i is presented to the RCA as the conjunction of the conditions of all nodes of c, together with $\neg u_i$.

Once the RCA has associated a goal to each sc_i (this set of goal is called G^1), he/she applies the selection σ3 (step 4), the hierarchy of RCs is updated such that:

$\forall g' \in G^2, G^2 = \sigma3(G^1), \exists RC' \in RC /$
$(g' = goal(RC')) \wedge (RC' \in OR^{trans}(RC))$.

Guiding rules C1, C2, and R2:

As these rules do not need any complex calculation, we do not provide any formal definition. The inclusion/exclusion checking in rule C1 relies on the use of the classical mathematical '$\subset$' function, by definition of the state as sets composed of elementary state components. The selection integration steps in rules C1, C2, and R2 are similar to the integration steps formulated above.

ACKNOWLEDGMENT

This work is partly funded by the Basic Research Action CREWS (ESPRIT No. 21.903). CREWS stands for Cooperative Requirements Engineering with Scenarios.

REFERENCES

[1] A.I. Anton, W.M. Mc Cracken, and C. Potts, "Goal Decomposition and Scenario Analysis in Business Process Reengineering," *Proc. Sixth Int'l Conf. CaiSE '94, Advanced Information Systems Eng.*, pp. 94–104, Utrecht, the Netherlands: Springer-Verlag, 1994.
[2] A.I. Anton, "Goal Based Requirements Analysis," *Proc. Second Int'l Conf. Requirements Eng., ICRE '96*, pp. 136–144, 1996.
[3] C. Ben Achour, "Guiding Scenario Authoring," *Proc. Eighth European Japanese Conf. Information Modeling and Knowledge Bases*, pp. 181–200, Ellivuori, Finland, May 1998.
[4] J. Bubenko, C. Rolland, P. Loucopoulos, and V. De Antonellis, "Facilitating 'Fuzzy to Formal' Requirements Modeling," *IEEE First Conf. Requirements Eng., ICRE '94*, pp. 154–158, 1994.
[5] A. Cockburn, "Structuring Use Cases with Goals," technical report, Human and Technology, HaT.TR.95.1, 84121, Salt Lake City, Utah, 1995. http://members.aol.com/acocburn/papers/usecases.htm
[6] B. Dano, H. Briand, and F. Barbier, "A Use Case Driven Requirements Eng. Process," *Proc. Third IEEE Int'l Symp. Requirements Eng. RE '97*, Anapolis, Md., IEEE CS Press, 1997.
[7] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal Directed Requirements Acquisition," *Science of Computer Programming*, vol. 20, nos. 1/2, pp. 3–50, Apr. 1993.
[8] A.M. Davis, *Software Requirements: Objects, Functions and States*. Prentice Hall, 1993.
[9] ELEKTRA consortium, "Electrical Enterprise Knowledge for Transforming Applications—Athena Deliverable: Initial Requirements for PPC," ELEKTRA Project Internal Report, http:/www.singular.gr/elektra, 1997.
[10] ELEKTRA consortium, "Esprit Program 7.1, Technologies for Business Processes, Best Business Practice Pilots," Elektra: Electrical Enterprise Knowledge for Transforming Applications. (Proj. no. 22927) http://www.singular.gr/elektra, Jan. 1997-June 1999.
[11] D. Filippidou and P. Loucopoulos, "Using Scenarios to Validate Requirements in a Plausibility-Centred Approach," *Proc. Ninth Int'l Conf. Advanced Information Systems Eng. CaiSE '97*, A. Olive, ed., Barcelona, Springer-Verlag, June 1997.
[12] I. Graham, *Migrating to Object Technology*. Addison-Wesley, 1995.
[13] G. Grosz, C. Rolland, S. Schwer, C. Souveyet, V. Plihon, S. Si-Said, C. Ben Achour, and C. Gnaho, "Modeling and Engineering the Requirements Engineering Process: An Overview of the NATURE Approach," *Requirements Eng. J.*, vol. 2, pp. 115–131, 1997.
[14] C.H. Holbrook, "A Scenario-Based Methodology for Conducting Requirements Elicitation," *ACM SIGSOFT, Software Eng. Notes*, vol. 15, no. 1, pp. 95–104, Jan. 1990.
[15] I. Jacobson, "The Use Case Construct in Object-Oriented Software Enginering," *Scenario-Based Design: Envisioning Work and Technology in System Development*, J.M. Carroll, ed., pp. 309–336, John Wiley & Sons, 1995.
[16] I. Jacobson, *Object Oriented Software Engineering, A Use Case Driven Approach*. Addison-Wesley, 1992.
[17] P. Kardasis and P. Loucopoulos, "Aligning Legacy Information System to Business Processes," *Proc. 10th Conf. Advanced Information Systems Eng., CaiSE '98*, pp. 8–12, Pisa Italy, June, 1998.
[18] J.C.S. do Prado Leite, G. Rossi, F. Balaguer, A. Maiorana, G. Kaplan, G. Hadad, and A. Oliveros, "Enhancing a Requirements Baseline with Scenarios," *Proc. Third IEEE Int'l Symp. Requirements Eng. RE '97*, pp. 44–53, Antapolis, Md., IEEE CS Press, 1997.
[19] P. Loucopoulos, "The F^3 (From Fuzzy to Formal) View on Requirements Engineering," Ingénierie des Systèmes d'Information, vol. 2, no. 6, pp. 639–655, 1994.
[20] P. Loucopoulos, V. Kavakli, and N. Prekas, "Using the EKD Approach, the Modeling Component," ELEKTRA project internal report, 1997.
[21] S.Nurcan, G. Grosz, and C. Souveyet, "Describing Business Processes with a Use Case Driven Approach," *Proc. 10th Int'l Conf. CaiSE '98*, Lecture Notes in Computer Science 1413, B. Pernici and C. Thanos, eds., Pisa Italy, Springer, June 1998.
[22] V. Plihon, J. Ralyté, A. Benjamen, N.A.M. Maiden, A. Sutcliffe, E. Dubois, and P. Heymans, "A Reuse-Oriented Approach for the Construction of Scenario Based Methods," *Proc. Int'l Software Process Assoc. Fifth Int'l Conf. Software Process (ICSP '98)*, Chicago, pp. 14–17, June 1998.
[23] K. Pohl and P. Haumer, "Modeling Contextual Information about Scenarios," *Proc. Third Int'l Workshop Requirements Eng.: Foundations of Software Quality REFSQ '97*, pp. 187–204, Barcelona, June 1997.
[24] C. Potts, K. Takahashi, and A.I. Anton, "Inquiry-Based Requirements Analysis," *IEEE Software*, vol. 11, no. 2, pp. 21–32, 1994.
[25] C. Potts, "Fitness for Use: the System Quality that Matters Most," *Proc. Third Int'l Workshop Requirements Eng.: Foundations of Software Quality REFSQ '97*, pp. 15–28, Barcelona, June 1997.
[26] N. Prat, "Goal Formalisation and Classification for Requirements Engineering," *Proc. Third Int'l Workshop Requirements Eng.: Foundations of Software Quality REFSQ '97*, pp. 145–156, Barcelona, June 1997.
[27] Rational Software Corp. *Unified Modelling Language Version 1.1.*, available at http://www.rational.com/uml/documentation.html, 1998.

[28] C. Rolland and C. Ben Achour, "Guiding the Construction of Textual Use Case Specifications," *Data & Knowledge Eng. J.* vol. 25, no. 1, pp. 125–160, P. Chen and R.P. van de Riet, eds., North Holland, Elsevier Science, Mar. 1997.

[29] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, M. Jarke, P. Haumer, K. Pohl, Dubois, and P. Heymans, "A Proposal for a Scenario Classification Framework," *Requirements Eng. J.*, vol. 3, no. 1, pp. 23–47, 1998.

[30] C. Rolland, S. Nurcan, and G. Grosz, "Guiding the Participative Design Process,. *Assoc. for Information Systems Americas Conf.*, Indianapolis, Ind., pp. 922–924, Aug. 1997.

[31] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modelling and Design*. Prentice Hall, 1991.

[32] S. Si-Said, "Guidance for Requirements Eng. Processes," *Proc. Eighth Int'l Conf. and Workshop Database and Experts System Application, DEXA '97*, Toulouse, France, Sept. 1997.

[33] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer, "Scenario Usage in System Development: A Report on Current Practice," *IEEE Software*, Mar. 1998.

[34] E. Yu and J. Mylopoulos, "Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering," *Proc. 27th Hawaii Int'l Conf. System Science*s, Maui, Hawaii, vol. 4, pp. 234–243, Jan. 1994.

Colette Rolland is currently professor of computer science in the Department of Mathematics and Informatics at the University of Paris-1, Panthéon Sorbonne. Her research interests lie in the areas of information modeling, databases, temporal data modeling, object-oriented analysis and design, requirements engineering., design methodologies, development process modeling, and CASE tools. She has extensive experience in participating in national and European research projects under the ESPRIT program (projects TODOS, BUSINESS CLASS, F3, NATURE, TOOBIS, ELKD, ELEKTRA, and CREWS) and conducting cooperative projects with industry. She is the French representative in IFIP TC8 on "Information Systems" and chair of the IFIP Working Group WG8.1.

Carine Souveyet is associate-professor in the Department of Mathematics and Informatics at the University of Paris-1, Panthéon-Sorbonne. Her research interests are directly related to method engineering, requirement engineering, temporal data modeling, and development process modeling. She is/was a work group leader in the F3 and TOOBIS ESPRIT projects and has played an active role in the CREWS ESPRIT project.

Camille Ben Achour received his BS and MS degrees in computer science from the Pierre and Marie Curie University in Paris, in 1993 and 1994, respectively. In December 1994 he joined the research group of Professor Rolland at the University of Paris-1, Panthéon-Sorbonne. He is currently a PhD candidate on the topic of scenario-based requirements engineering. His research interests include requirements engineering, conceptual design, database theory, deductive databases, and object-oriented databases.