Crews Report Series 98 - 23

L'ECRITOIRE a tool to support a goal-scenario based approach to requirements engineering

M. Tawbi, C. Souveyet, C. Rolland

University of Paris1- Sorbonne
CRI
90 rue de Tolbiac 75013 Paris, France.
{tawbi, souveyet,rolland}@univ-paris1.fr

Submitted to Information and Software Technology Journal (ed. M. Shepperd) Elsevier Science Publishers

L'ECRITOIRE a tool to support a goal-scenario based approach to requirements engineering

Mustapha Tawbi, Carine Souveyet, Colette Rolland
email {tawbi, souveyet,rolland}@univ-paris1.fr
Centre de recherche en Informatique (CRI), University of Paris1 Panthéon Sorbonne,
90 rue de Tolbiac 75013 Paris, France.

Abstract

The paper presents 'L'ECRITOIRE' a tool which supports the goal-scenario based approach for requirements engineering developed within the CREWS project. The approach tightly couples goal modeling and scenario authoring to elicit system requirements The approach is supported by guidelines encapsulated in modular method chunks that are assembled in different ways called method paths. A method path is a situated 'good practice' of goal modeling and scenario authoring. The paper presents one of the CREWS_L'ECRITOIRE method paths and illustrates it use through a L'ECRITOIRE session.

1. INTRODUCTION

Scenario have recently gained attention in the field of Requirement Engineering (RE) A *scenario* is 'a possible behavior limited to a set of purposeful interactions taking place among several agents' [15,16]. It describes a desirable functionality of a system under design, and thus, help identifying requirements.

Goal modeling is another alternative way to facilitate requirements elicitation[5]. Our experience is that it is difficult for domain experts to deal with the fuzzy concept of a goal [2, ,17,8, 9]. It is often assumed that systems are constructed with some goals in mind [6]. However, practical experiences [1, 7] show that goals are not given and therefore the question as to where they originate from [1] acquires importance. In addition, enterprise goals which initiate the goal discovery 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.

In the ESPRIT CREWS [1] project, we propose to do this by *combining goal driven approaches* for Requirements Engineering (RE) *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. These two main activities had been dealt in [3] for the authoring aspect and in [13] for scenario exploration to discover goals.

The process which combines goal modeling and scenario authoring is a complex one that we want to support by guidelines implemented in enactable method chunks organized in methods paths[14]. Each path corresponds to a given flow of steps, each being supported by a method chunk. Method chunks and method paths are implemented in the software tool L'ECRITOIRE which is the subject of this paper.

The paper is organized as follows. First, we introduce in section 2, the notion of a requirement chunk which is a pair <goal, scenario> The process combining goal discovery and scenario

[1] This work is partly funded by the Basic Research Action CREWS (ESPRIT N°21.903). CREWS stands for Cooperative Requirements Engineering With Scenarios.

authoring is introduced in section 3 together with the notions of method chunks and method paths. The method path considered in this paper is presented in section 4 and illustrated through a session using 'L'ECRITOIRE' tool in section 5. Finally, we draw some conclusions in section 6 .

2. THE NOTION OF A REQUIREMENT CHUNK

At the core of our approach is the Requirement Chunk (RC), defined as the pair <goal, scenario> 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 in which the goal can be achieved[13]. Requirements chunks can be assembled together either through *composition* and *alternative* relationships or through *abstraction* relationships. The former lead to AND and OR structure of RCs whereas the latter leads to the organization of the RC model as a hierarchy of chunks of different granularity. (Figure 1) sumps up these relationships, using OMT notations [4].

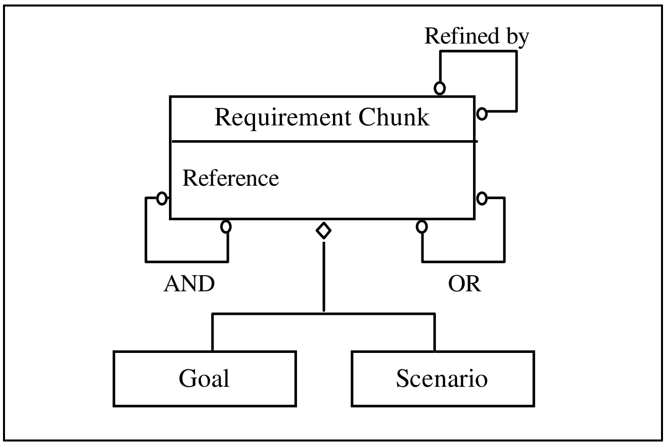

Figure 1:the requirement chunk notion

3. THE PROCESS VIEW

The requirement chunk is the central part of the goal discovery process (figure 2). Each step in this process consists of two phases :

(1) *scenario authoring*
(2) *goal discovery*

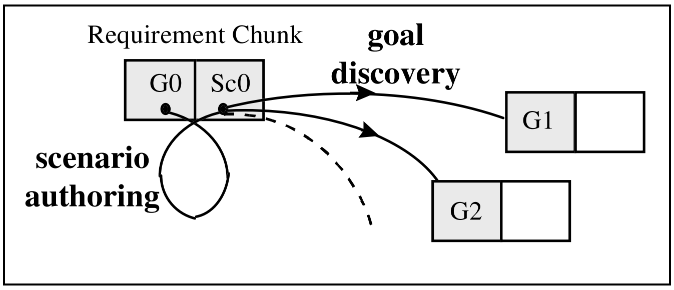

Figure 2: Overview of the discovery process

Both are supported by rules, (1) authoring rules and (2) discovery rules. We assume scenarios to be textual and use authoring rules [12] to provide style and contents guidelines as well as linguistic devices for analysis, disambiguation and completion. The linguistic devices are based on a case grammar and case patterns. Details can be found in [12]. Discovery rules are of three types to respectively help discovering *ANDed* , *ORed* and *Refined* goals given a requirement chunk RC.

Rules are encapsulated in *method chunks* organized in *method paths*.

A *method chunk is contextual :* it is a component embedding guidelines to achieve a specific intention applicable in a given situation. As shown in (figure 3), a method chunk has an interface and a body.

The chunk interface is a pair <situation, intention>. Where the situation is the part of the product required as a pre-requisite for the fulfillment of the intention. The intention is the goal to be achieved in that particular situation. For example the following interface <situation=(*Goal* : *g*), intention=*Elicit scenario using CREWS authoring guidelines*> expresses the fact that a *Goal* is a required product part for achieving the intention to elicit a *Scenario.* The interface <situation=(*Scenario : s)*, intention=V*erify scenario by checking the vocabulary*> characterizes a method chunk to guide the *verification of a scenario by checking the vocabulary* which is applicable in a situation where a *scenario* has been already elicited. As shown by these two examples, the interface of a method chunk situates its context of application.

The body chunk contains the guidelines to be applied when the chunk is actually used. For capturing a large range of method chunks, the body is either formal or informal. The former implies to define method chunk as proposed by the Nature process formalism [10,11]. Detailed examples of such a description can be found in [12,18]. The latter describes method chunk in natural language.

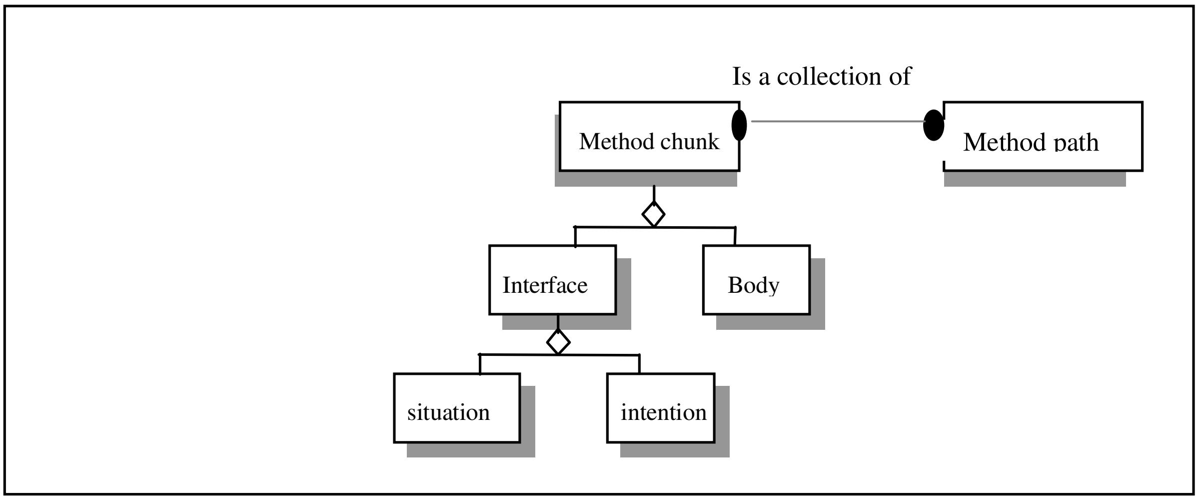

Figure 3: Overview of a method chunk

A method chunk alone is not useful. Methods chunks must be grouped in a *method path* to support an entire process [14]. We define a method path as a *possible route among method chunks*. Our view is that several method paths can be defined to be applied in specific project situations, each corresponding to a specific grouping of method chunks. Thus we adopt a modular view of method construction : a chunk is seen as a module which is self contained, has a well defined interface and therefore, can be assembled to other chunks modeled according to the same template We have currently implemented a repository comprising 18 method chunks which are grouped in four different paths. In the next section we present the method path selected for this paper.

.

4. The CREWS method path MP1

The method path used in this paper is visualized in figure 4. It is made of an iterative cycle which includes four main intentions to fulfill.

1) Discover goals from a given goal
2) Elicit scenario
3) Conceptualize scenario
4) Discover goals from scenario

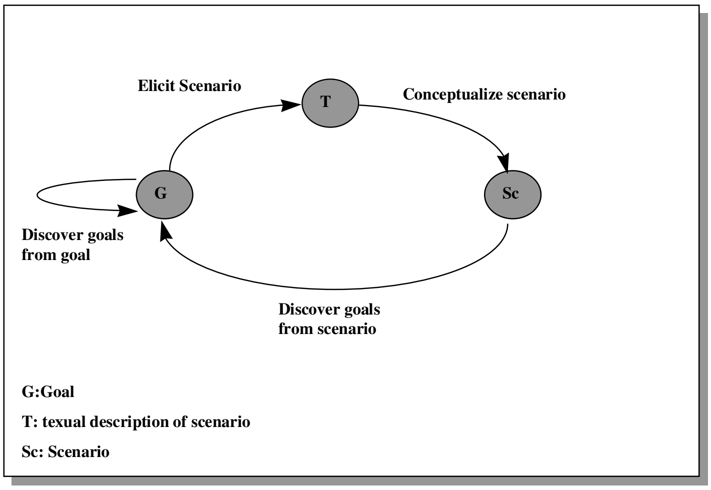

Figure 4:the method path MP1

This process takes as input, a goal, produces a RC by authoring scenarios attached to this goal and supports the discovery of new goals which in turn can each, be input of a new loop. The fulfillment of each of these intentions is supported by method chunks. These method chunks and the associated path are implemented in the L'ECRITOIRE tool to support the flow of decision making which composes the process as shown in figure 4.

Furthermore, the method path focuses on two types of requirements chunks namely, *system interaction RCs* and *system internal RCs*. As a consequence, the path is initiated by a given service goal (see figure 5) and conducts the previous loop at two levels of abstraction : the *system interaction* level and the *system internal level*

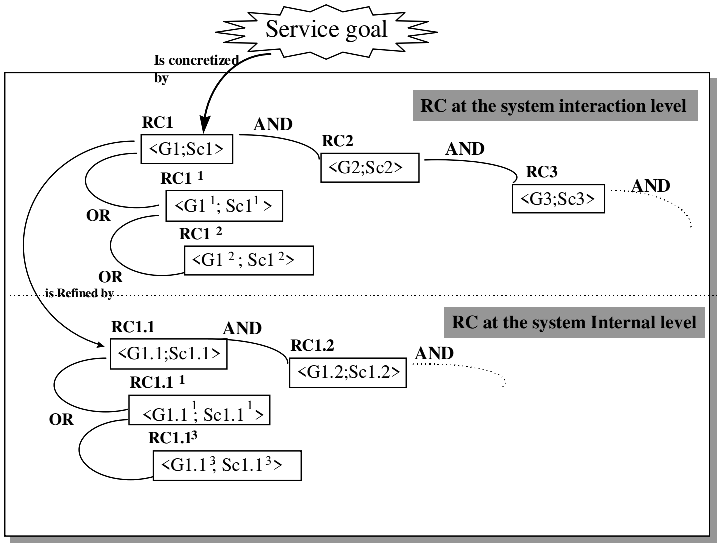

Figure 5 :the two levels of abstraction considered in MP1

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. A *system interaction RC* captures one way of providing a service . It couples a *service goal* and a *system interaction scenario*. A *service goal* expresses a manner of providing a service. The associated *system interaction scenario* describes a flow of interactions between the system and its users which allows the users to fulfill the service goal.

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 system internal actions that involve system objects but may require external objects such as other systems. System interactions are refined in system internal chunks and new ones are added. A *system internal RC* details one possible way in which the system may internally perform an interaction identified in a system interaction scenario at the previous level. It combines a *system goal* and a *system internal scenario*. A *system goal* expresses a manner to perform an action identified in a system interaction scenario.

During the method path enactment, the requirement chunk author (RCA) starts with a service goal in mind. Then, the loop introduced in figure 4 is performed at the system interaction level first and then, at the system internal level. The result is a RCs structure organized at two level of abstraction with, for each level chains of ORed and ANDed requirements chunks.

We detail in turn each of the four key intentions of the path (see figure 4) and shall illustrate the tool support in the following section. All examples correspond to the ATM case study.

4.1 Discover goals from a given goal :

This intention is supported by the chunk :
« situation : (goal), intention : (discover goals from goal) ».

The corresponding guideline aims at supporting the discovery of new goals from one a given goal. Let us consider the following example from the ATM case : *'Provide cash with a card based ATM'*

The guideline uses a predefined template for goal formalisation and suggests to the requirement chunk author (RCA) to provide alternative values of the goal parameters. This might lead in our example to propose, 'account balance information', 'money transfer facilities' as alternatives to 'cash withdrawal'. and 'a code based ATM ' as an alternative mean to « a card based ATM ».

All possible combinations of parameters are computed and then, presented to the RCA parameter value-wise. Finally the RCA is invited to select from the set of generated goals a sub-set of new goals of interest. *'Provide cash from account using a card based ATM'* and *'Provide account balance information from account using a code based ATM'* are two examples of goals discovered from the initial goal *'Provide cash with a card based ATM'*.

4.2 Elicit scenario

There are two method chunks which support the fulfillment of this intention :

- <situation=(Goal), intention=Write textual scenario using CREWS authoring guidelines>,
- <situation=(LN text), intention=Check spelling>,

The first one provides guidelines to help the author writing the scenario according to its type with respect to its intended contents. Guidelines are of two kinds : style and contents ; the former deal with the wording of the text and the latter focuses on the nature of the scenario contents.

The second one provides spelling facilities that we borrowed from Microsoft WORD.

4.3 Conceptualize scenario

This intention is associated to four chunks which support the transformation of an initial textual scenario into a well structured text, free of any ambiguity and linguistically completed. The four chunks correspond to the following sub_intentions :

- to check the initial text against the domain glossary,
- to analyse the textual scenario using a case grammar,
- to verify the linguistic completeness of the scenario , and
- to generate the final NL structured text.

We give in the following a short explanation to each one of these chunks.

4.3.1 Check the initial text against the domain glossary

The interface of this chunk is :

<situation=(LN text), intention= Check the initial text against domain glossary>,

The chunk aims at ensuring that the terms used in the scenario conform to a domain glossary. This requires that the domain glossary has been defined. This can be done either *a priori* or *'on the fly'*. The former is supported by the following method chunk :

<situation=(goal), intention = Define domain glossary>,
The latter is embedded in the above mentioned chunk for controlling the vocabulary.

The idea behind the construction of this glossary is based on the fact that common words are frequently used when dealing with a specific domain. In the ATM domain, words like « user », « card », « cash » are often invoked. The management of a glossary for a project came out of the practice survey performed by the CREWS team in industries[].

The chunk verifies the conformance of terms used in the scenario with glossary. It also searches if terms in the text are synonyms of terms in the glossary and asks the RCA to select the preferred one but to avoid redundancy.

4.3.2 Analyse the textual scenario using a case grammar

The interface of this chunk is :
<situation=(LN text), intention=Analyse the textual scenario using a case grammar>,

This chunk uses linguistic devices to analyse the semantic contents of the textual scenario and represent it as a collection of instantiated linguistic cases patterns [12]. There are two types of semantic patterns, **clause** and **sequence patterns.** The former provide the semantic of atomic actions such as*« the user inserts his card in the ATM »* whereas the latter provide the semantic of complex actions such as *« the user inserts his card in the ATM , then the ATM gives a prompt for code to the user »*

A verb is the main concept of a **clause pattern** . A verb is the center and we attach to it the other parameters of the sentence. We classify verbs in two categories : *action verbs* and *communication verbs.*
Action verbs care used in simple actions like «check » or « validate ». An action verb requires an agent parameter which is the pronoun of the verb and an object parameter which is the subject of the verb. Then, the sentence « the ATM checks the card validity » will be represented as the instantiated pattern :Action(checks)[agent : the ATM ; object : the card validity]

Communication verbs are used in actions where agents exchange objects which can be physical objects like 'a card' or information objects like 'a message 'or 'a prompt'. For example in the action « the user inserts the card in the ATM » 'inserts is a communication verb and 'the card' is a physical object. A communication verb needs also two parameters : a source ('the user') and a destination('the ATM'). So« the user inserts the card in the ATM » corresponds to the instantiated pattern :Communication (input) [Agent : the user ; Object : the card ; Source : the user ; destination : the ATM] translated to semantic patterns representation.

The purpose of **Sequence patterns** is to compose simple actions in complex ones. We classify complex actions in four categories :

Sequence : « the user inserts his card in the ATM, then the ATM gives a prompt for code to the user »
Condition **:** « If the card is valid then a prompt for code is given by the ATM »
Iteration : « for each transaction the ATM sends a notification to the bank »
Concurrency : «The ATM returns the card while a receipt is printed »

The chunk uses four format of sequence patterns modeling the four types of sequences. **Sequence [before : Action1 ; after :Action2]** corresponds to the sequence type and **Constrained [condition : Action1 ; constrained : Action2]** corresponds to the condition type.

Obviously these sequence formats are recursive ones, so action1 and action2 are semantic patterns which can be of any of the types described above.

4.3.3 Verify the linguistic completeness of the scenario

The interface of this chunk is as follows:
<situation=(instantiated semantic patterns),
intention= verify the linguistic completeness of the scenario>,

This chunk aims to clarify and complete a textual scenario. For this purpose, the chunk applies two parallel strategies. The first one aims to complete interaction statements whereas the second one aims to detect and disambiguate anaphoric references .

To illustrate the former, let us consider the action *« a prompt for code is given »* and its corresponding instantiated pattern
communication(give) [Agent : ? ; object : a prompt for code ; Source : ? ; Destination : ?]
The completeness strategy consists in adding all missing parameters in the instantiated patterns. The RCA in the previous example is asked to replace every '?' by a glossary term This leads to the completed sentence :

« a prompt for code is given by the ATM to the user »

Anaphoric references are detected and replaced by non ambiguous terms. For example in the action *« the user inserts his card in the ATM »* which corresponds to
communication (insert)[Agent : the user ; Object : his card ; Source : the user ; Destination : The ATM].
the anaphoric reference 'his' in 'his card' is detected and the RCA is asked to replace 'his' by a glossary term let say, the 'user' and the action is rephrased as
« the user inserts the user's card in the ATM ».

4.3.4 Map onto structured NL text

The interface of this chunk is :
<situation=(analysed and completed text), intention= map onto structured NL text >,

The role of this chunk is to produce a formal scenario description in structured natural language (See L'ECRITOIRE session for an example). The mapping uses a correspondence between every pattern and its surface structure.

4.4 Discover goals from scenario

The interface of this chunk is as follows:

<situation=(scenario description), intention= discover goals from scenario >,

The body includes three different strategies to discover new goals. Each of them exploit one of the possible type of relationship which exists between chunks (see section 1). We introduce these strategies with examples , for more detail see [13] :

a) S1 :Discover *refined goals* to the goal G of a given RC:

This strategy looks to every interaction in a scenario belonging to the level i of abstraction as a goal at level i+1. The chunk scans every interaction in a scenario and asks the RCA to confirm or infirm the fact that the interaction should be regarded as a goal. In the positive case, the RCA is asked to rephrase the selected action as a goal statement. The discovered goal belongs to a requirement chunk which refines. the initial RC. For example, given the atomic action « *The ATM report the cash transaction to the bank* », the RCA can decide to treat as a goal named « *Report the cash transaction to the bank* »

b) **S2 :Discover *ORed goals* to the goal G of a given RC**

This strategy aims to identify ways to achieve a given service goal but in different manners. S2 builds a graph representing all the possible paths of actions already identified in the scenario of the requirement chunk under investigation.

Every path is characterized by *zero* to *n* nested flow conditions. For example, in the scenario associated to the goal *'Withdraw cash from the ATM in a normal way',* there are four nested flow conditions (see example in section 4) :

1- If the card is valid
    2-If the code is valid
        3- If the amount is valid
            4- If a receipt is asked by the bank customer to the ATM

S2 computes all the combinations of negative conditions that should be investigated as possible missing paths. S2 generates four cases in our example :

(1)	*card is not valid*
(2)	*(card is valid) & (code is not valid)*
(3)	*(card is valid) & (code is valid) & (amount is not valid)*
(4)	*(card is valid) & (code is valid) & (amount is valid) & (receipt is not asked)*

These cases help to discover new goals such as *'Withdraw cash with error correction.* These goals belong to RCs that are all related to the initial RC through an OR relationship'

a) **S3 : Discover *ANDed* goals from a given RC**

The strategy S3 uses interactions among objects which are resources. For example, in the *'Withdraw cash from ATM in a normal way'* scenario, the re are three resources : the ‘card’, the ‘cash’ and the ‘receipt’.

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. The pairs identified in our example are the following :

- card : (‘the bank customer inserts the card’; ‘the ATM ejects the card to the bank customer’),
- receipt : (? ; ‘the ATM prints a receipt to the bank customer’), and
- cash : (? ; ‘the ATM delivers cash to the bank customer’).

Every incomplete pair originates a new goal. There are two incomplete pairs, noted with a question mark. Thus, two new goals are suggested, that the RCA accepts and names :

- *'Fill in the ATM with receipt paper'*, and
- *'Fill in the ATM with cash'.*

These two goals belong to RCs that are ANDed to the initial one.

5. A scenario of use with the L’ECRITOIRE tool

This section is a usage scenario of the MP1 of the CREWS approach . The scenario shows how the requirement chunk author (RCA) interacts with the L’ECRITOIRE tool to get the support, guidance and help he/she wants. The scenario corresponds to one path through the system interaction and system internal levels. It illustrates a top down approach starting from a service goal and refining it through a number of steps to reach a complete specification of the system requirements.

At the begin of the session , ‘L’ECRITOIRE’ asks the RCA to define his/her domain glossary. This is done either by writing a new glossary or by restoring one from the repository of domain glossaries. Figure 6 shows the tool interface of the glossary construction. As shown the glossary is structured into lists : of verbs , agents and objects relevant for the application domain. A ‘Check Vocabulary ‘ button is also offered to remove redundancy related to the use of synonyms in the glossary. For example, the tool detects that the terms ‘cash’ and ‘money’ are synonyms in the object list and it proposes to the RCA to remove one of the two .

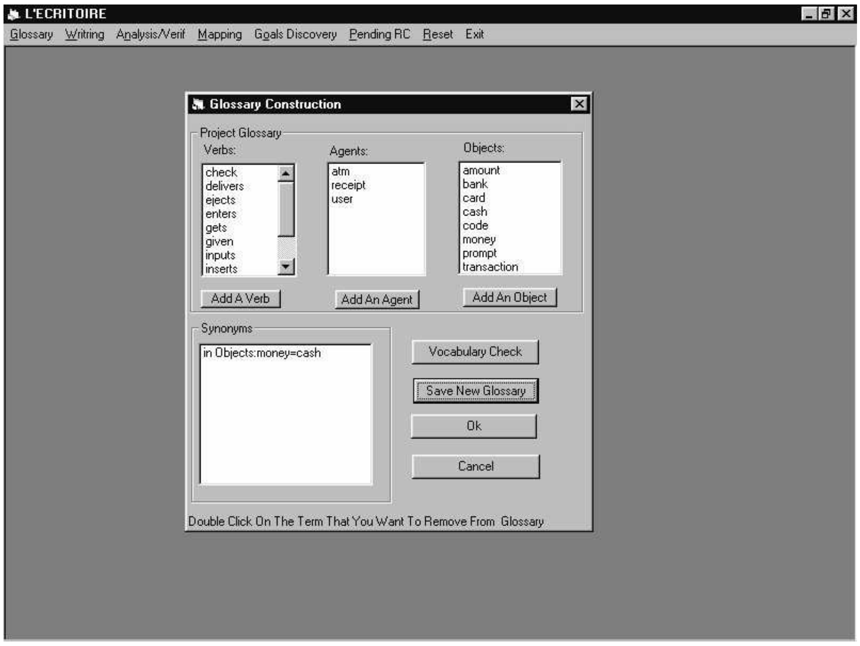

Figure 6:the glossary construction interface

As explained in the previous section, in the MP1 path, the RCA starts with a service goal in mind. This is *'Withdraw cash from the ATM'* in our case. The next step is to author a scenario which illustrates one way to achieve this goal. As shown in figure 7 the interface named 'Writing' serves this purpose.

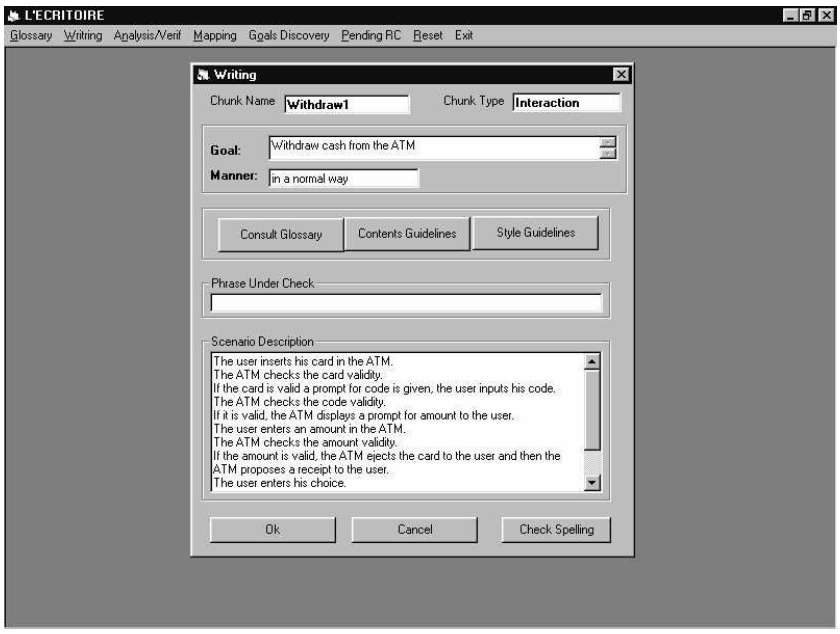

Figure 7: the Write interface

As it is visible in figure 7, the tool offers two buttons to select the type of guidelines that the RCA wants to use. : Contents or Style guidelines which are specialized for each type of RC (Interaction and Internal). As indicated in the bottom of the interface a 'Check spelling' functionality is provided which is identical to the one offered by Microsoft Word.

The underneath figure 8 reproduces the text written by the RCA in the window presented in figure 7.

The user inserts his card in the ATM.
The ATM checks the card validity.
If the card is valid , a prompt for code is given, the user inputs his code.
The ATM checks the code validity.
If it is valid , the ATM displays a prompt for amount to the user.
The user enters an amount.
The ATM checks the amount validity.
If the amount is valid , the card is ejected and then the ATM proposes a receipt to the user.
The user enters his choice.
If a receipt was asked, the receipt is printed but before the ATM delivers the cash.

Figure 8 :the initial textual scenario

At the end of this step, the intention '*Elicit scenario'* of the illustrated method path MP1 is achieved. The RCA is invited now to consider the *'Conceptualize scenario'* intention.

As shown i figure 9 this intention is supported in the L'ECRITOIRE by the interface called 'Analysis and Verification' (Figure 9)

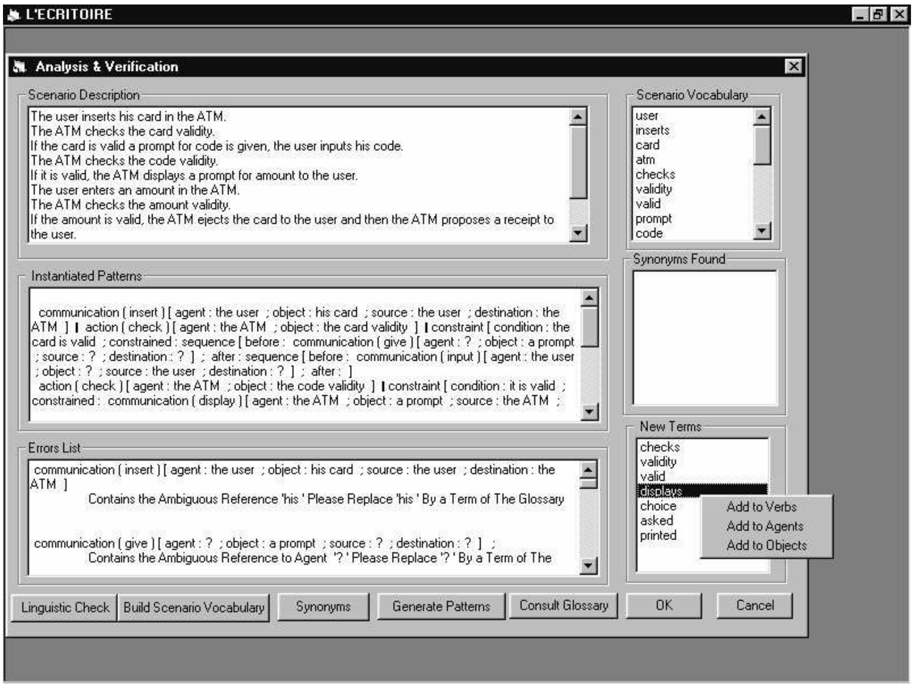

Figure 9: the Analysis and Verification interface

The textual scenario written during the previous step is displayed in the text zone of the top left of the interface. The two lists at the top ('Scenario vocabulary') and the bottom right ('New Terms') are generated with the 'Construct Glossary' button. The former corresponds to the terms used in the scenario, whereas the later corresponds to scenario terms which are not parts of the glossary. The RCA is given the freedom to choose terms from this list and to add them to the glossary if they are not synonyms of already existing terms. The list in the middle right (‘synonyms found’) is generated by the Button ‘Synonyms’ and corresponds to synonyms of glossary terms used in the scenario. The RCA is asked to replace the synonym in the scenario by its corresponding terms of the glossary. In our example case there are no synonyms.

The results of the linguistic analysis is shown in the middle left text. They take the form of ‘instantiated patterns’ and are generated by pushing the ‘Generate Patterns’ button

The ‘Errors list’ text zone is generated with the button ‘Check linguistic’ which triggers the chunk detecting ambiguities in the instantiated semantic patterns. In this method path, as the aim is to obtain complete and detailed system specification, the RCA is asked to correct all detected ambiguities. At the end of this activity, the scenario description is as follows (modifications are shown in bold).

The user inserts **a card in the ATM.** The ATM checks the card validity. If the card is valid a prompt for code is given **by the ATM to the user**, the user inputs the code **in the ATM**. The ATM checks the code validity. If **the code is valid**, the ATM displays **a prompt for amount to the user**. The user enters an amount **in the ATM**. The ATM checks the amount validity. If the amount is valid, **the ATM ejects the card to the user** and then the ATM proposes a receipt to the user. The user enters **the user's choice** in the ATM. If a receipt was asked the receipt is printed **by the ATM to the user** but before the ATM **delivers the cash to the user.**

Figure 10:the disambiguated textual scenario

As the scenario is disambiguated now, **‘L’ECRITOIRE’** offers to the RCA the facility for mapping the text onto its final version i.e. the NL structured description . This is performed within the ‘Mapping’ interface (figure 11).

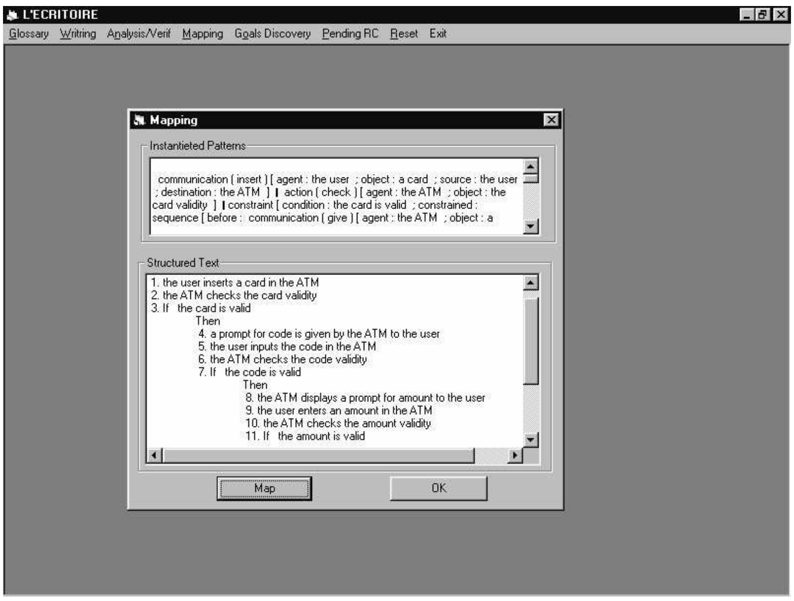

Figure 11 : the Mapping interface

In the upper text the interface displays the ‘Instantiated Patterns’. The final scenario description is obtained through the execution of the corresponding method chunk by clicking on the button ‘Map’ The results is shown in figure 11 and reproduced in figure 12 for sake of readability.

1. the user inserts a card in the ATM
2. the ATM checks the card validity
3. If the card is valid
   Then
   4. a prompt for code is given by the ATM to the user
   5. the user inputs the code in the ATM
   6. the ATM checks the code validity
   7. If the code is valid
      Then
      8. the ATM displays a prompt for amount to the user
      9. the user enters an amount in the ATM
      10. the ATM checks the amount validity
      11. If the amount is valid
          Then
          12. the ATM ejects the card to the user
          13. the ATM proposes a receipt to the user
          14. the user input the user's choice
          15. If a receipt was asked
              Then
              16. the ATM delivers the cash to the user
              17. the receipt is printed by the ATM to the user

Figure 12: the final scenario description

At this point , the RCA has achieved the ‘Conceptualize Scenario’ intention, and he/she is invited now to proceed with the next intention of the path MP1 :‘Discover Goals from Scenario’ intention. This is supported by the ‘Goal Discovery’ interface of the L’ECRITOIRE which is displayed in figure 13.

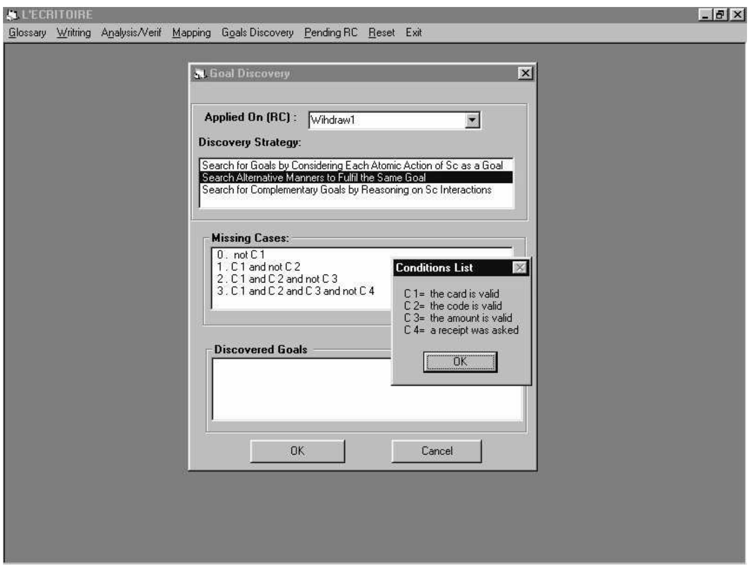

Figure 13: the discover goals from scenario interface

The RCA can select a specific goal discovery strategy from the zone list provided on the interface ('Discovery strategy zone). Let assume that he/she chooses first the S2 strategy. As presented in section 3 the method chunk enactment computes the missing paths in the scenario and asks the RCA to rephrase them as new goals if he/she finds them relevant. In our case, the RCA writes ***'Withdraw cash with an error correction phase*** (Figure 14).

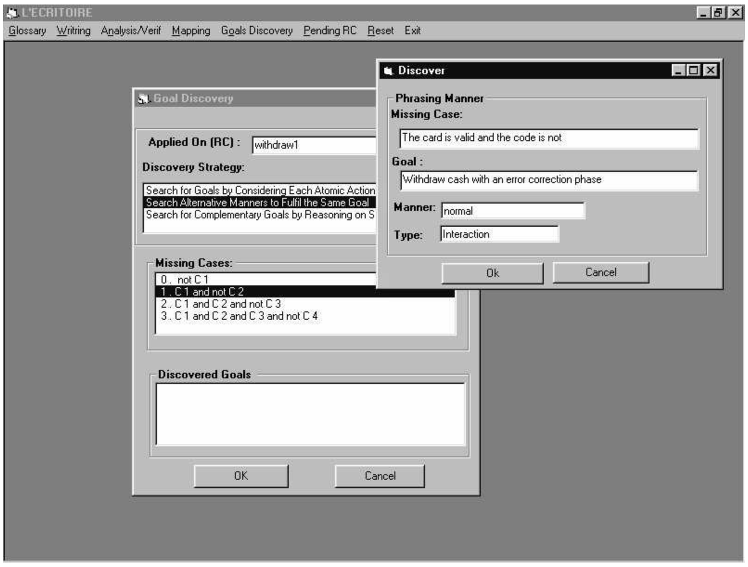

Figure 14: discover goal applying S2 strategy

Let assume that the RCA applies S2 once, and S1 twice. This will lead. to introduce the two new goals :
Check the card validity'and
Check the code validity'
which are two system internal goals.

At this point our session reaches to its end. When the RCA will launch a new session,'L'ECRITOIRE' will displays a list of pending RCs corresponding to the set of discoverd goals which have not been authored, yet (figure 15).

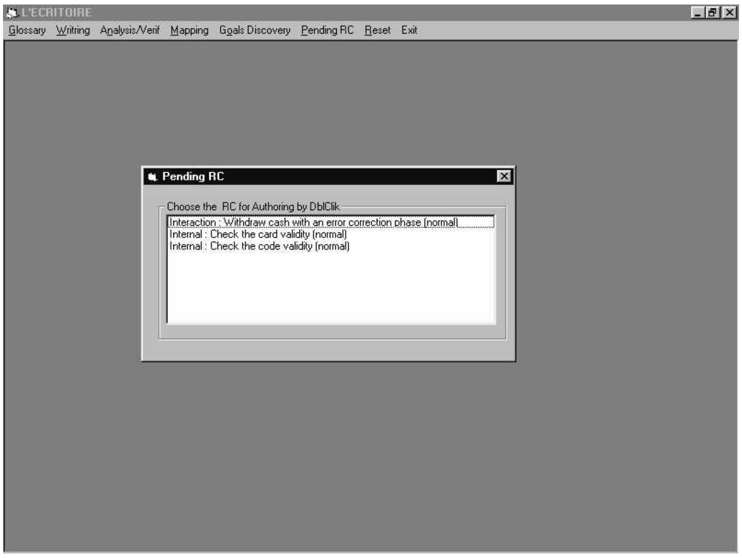

Figure 15: The list of goals have not been authored

6. Conclusion

We presented the CREWS_L'ECRITOIRE approach to guide requirements elicitation by combining goal modeling and scenario authoring. The originality of the guidance provided by the tool called L'ECRITOIRE lies in the modular description of method guidelines and their flexible assembly into method paths. Guidelines are encapsulated in method chunks which are cohesive modules that can be coupled in different ways to constitute a method path well suited to a specific project situation. The paper illustrated one path with the ATM example. The tool is itself implemented with software chunks and software paths, thus the conceptual assembly of method chunks in a method path can be easily implemented in L'ECRITOIRE. The approach has been validated with examples such as the ATM case study. We are currently working in two directions (a) validating the approach by considering complete cases and diversifying cases and (b) defining new paths of 'good practice' in requirements engineering using our goal_scenario based approach.

7.Références

[1] A.I. Anton, *Goal based requirements analysis*. Proceedings of the 2nd International Conference on Requirements Engineering ICRE'96, pp. 136-144, 1996.

[2] J. Bubenko, C. Rolland, P. Loucopoulos, V De Antonellis, *Facilitating 'fuzzy to formal' requirements modelling*. IEEE 1st Conference on Requirements Enginering, ICRE'94 pp. 154-158, 1994.

[3] C. Rolland, C. Ben Achour, *Guiding the construction of textual use case specifications*. Accepted to Data and Knowledge Engineering Journal, 1997.

[4] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *''Object-oriented modelling and design''*. Prentice Hall, 1991.

[5] C. Potts, *Fitness for use : the system quality that matters most.* Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'9 , Barcelona, pp. 15-28, June 1997.

[6] A.M. Davis, *''Software requirements :objects, functions and states''*. Prentice Hall, 1993.

[7] ELEKTRA consortium, *Electrical Enterprise Knowledge for Transforming Applications - Athena deliverable : initial requirements for PPC.* ELEKTRA Project Internal Report, 1997.

[8] P. Loucopoulos, V. Kavakli, N. Prekas, *Using the EKD approach, the modelling component.* ELEKTRA project internal report, 1997.

[9] C. Rolland, S. Nurcan, G. Grosz, *Guiding the participative design process.* Association for Information Systems Americas Conference, Indianapolis, Indiana, pp. 922-924, August, 1997.

[10] V. Plihon, C. Rolland : « Modeling Ways of Working », Proc. Of the 7th International Conference on advanced information Systems Engineering, CAISE'95, Springer Verlag 95.

[11] C Rolland, G. Grosz : « A General Framework for describing the requirement engineering process », IEEE conference on Systems, Man & Cyberenetics, CSMC'94, San Antonio, Texas 1994.

[12] C. Rolland, C. Ben Achour : « guiding the construction of textual use case specifications », accepted to Data Knowledge Engineering Journal.

[13] : C. Rolland, Carine Souveyet, Camille Ben Achour : « Guiding Gaol Modelling Using Scenario » , Proposed to TSE journal, TSE special issue on Scenario Management 1998.

[14] :C . SOUVEYET, M. TAWBI : « Process centred Approach for developing tool support of situated methods », Proposed to DEXA'98, 9th International Conference and Workshop on. Database and Expert Systems Applications. August1998, Austria, Vienna

[15] J.M. Caroll, The Scenario Perspective on System Development, in J.M. Carroll (ed.), Scenario-Based Design: Envisioning Work and Technology in System Development (1995).

[16] R.L. Mack, Discussion : Scenarios as Engines of Design, in John M. Carroll (ed.), Scenario-Based Design: Envisioning Work and Technology in System Development (John Wiley and Sons, 1995) 361-387.

[17] P. Kardasis, P. Loucopoulos, *Aligning legacy information system to business processes.* Submitted to CAiSE'98, 1998.

[18] C. Rolland , V. Plihon : « Using generic chunks to generate process model fragments », Proc. On the 2th Conference on requirements engineering, ICRE'96, Colorado Springs,1996