CREWS Report Series 98 - 24

A PROPOSAL FOR IMPROVING THE QUALITY OF THE ORGANISATION OF SCENARIOS COLLECTIONS

Camille Ben Achour[†‡], Colette Rolland[‡], Carine Souveyet[‡]

‡ CRI Université de Paris 1 - Sorbonne
90 rue de Tolbiac
75013 Paris - France

† LIP6 - Université Université de Paris 6 - P.&M. Curie
4, place Jussieu
75005 Paris - France

{camille, rolland, souveyet}@univ-paris1.fr

Appeared in the proceedings of the 4th International Workshop on Requirements Engineering Foundation for Software Quality. E. Dubois, A. Opdahl, K. Pohl (Eds). June, Pisa, Italy.

A proposal for improving the quality of the organisation of scenarios collections[1]

Camille Ben Achour†‡, Colette Rolland‡, Carine Souveyet‡

‡ CRI Université de Paris 1 - Sorbonne
90 rue de Tolbiac
75013 Paris - France

† LIP6 - Université Université de Paris 6 - P.&M. Curie
4, place Jussieu
75005 Paris - France

{camille, rolland, souveyet}@univ-paris1.fr

Abstract

The study and practice of scenarios raise the question of managing the combinatorial explosion of the number of scenarios. This article proposes a set of relationships to organise a collection of scenarios. Three types of relationships are proposed : OR, AND, and Refinement. These relationships are generic and can be used to manage any kind of scenarios. Using them leads to build a structured network of scenarios. In addition, to guide the construction of 'quality' networks of scenarios, the paper presents quality properties and heuristic guidelines. The approach is illustrated by the CREWS approach.

Keywords : Scenario, Requirements Engineering, Combinatorial Explosion

1. Introduction

Scenarios are popular in the domains of Human Computer Interactions (HCI), Software Engineering (SE), Requirements Engineering (RE), Strategic Management (SM) or Decision Theory (DT). Scenarios help focus discussion on usability issues, on understanding design goals, on setting strategic visions of a system, assisting in the identification of system functionalities, etc. Leading researchers in scenario based approaches see the problem of the 'combinatorial explosion' as one of the two most preoccupying issues [7]. scenarios are partial views, and collections of scenarios are necessary to obtain more complete views. Reports from industrial practitioners raise the same issue. For example, the CREWS survey of scenario practices in European Companies shows that the RE process relies on high volumes of scenarios [8] which are not well managed today.

The problem has been tackled by a few authors, for instance by Jacobson [6] who proposes the 'uses' and 'extend' relationships as a means to relate scenarios one another. However these relationships are syntactical and they are specific to the Jacobson's approach. Tackling the problem with a different approach, Cockburn [3] proposes to use goals to track scenarios. From this combination, he deduces the relationship of ‘variation’ between scenarios. Other relationships could also be considered.

In this paper we propose three kinds of relationships : AND, OR and Refine. Despite these relationships have been defined within the CREWS project [15], we believe that they are generic and can be used to organise collections of scenarios of different kinds : textual, graphical, animated, functional or not, etc. [14]. The AND / OR / Refine relationships are semantic relationships which relate scenarios according to the nature of their contents. The solution proposed has been applied on examples [13, 15], and on real case studies [10].

---

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

The paper is organised as follows. The next section is an overview of the CREWS approach for scenarios based requirements engineering. Section 3 addresses the lessons learned from the CREWS approach practice and presents our proposal for structuring collections of scenarios. Section 4 proposes guidelines to support a quality structure of scenarios collections.

2. The CREWS approach

The CREWS approach to requirements engineering combines the use of the two concepts of goal and scenario into the notion of a requirement chunk.

A *goal* is defined as *'something that someone hopes to achieve in the future'* [15]. By 'someone', we mean the *stakeholder* who elicits and conceptualises the goals. A stakeholder can be the user of the designed system, the owner of the future system, the requirements engineer, etc. We view goals as the stakeholders' intentions regarding the system under design.

A *scenario* expresses a possible way in which the goal can be achieved. In the CREWS perspective, a scenario is defined as *a 'possible behaviour limited to a set of purposeful interactions taking place among several agents'* [15]. A scenario includes one or several *actions*, but, by definition, the combination of actions in a scenario describes a unique path from an initial to a final state.

A *requirement chunk* (RC) is a pair <goal, scenario>.

The CREWS approach addresses the *authoring of scenarios* and the *discovery of goals* [1,15]. Both are guided by rules. Goal discovery is guided by situated rules. Each rule proposes a way to fulfil the stakeholder's discovery intention for a well-defined situation. Scenario authoring is composed of two steps, scenario writing and scenario correction. Scenario writing is guided by textual guidelines. Once the stakeholder has written a scenario, he can correct it being guided by enactable rules that identify suspect situations and propose solutions to correct the scenario.

These rules are encapsulated into *method chunks* that are organised into *method paths. A method chunk* helps transform an initial situation into a result according to a certain intention.

Method paths propose strategies for combining the application of method chunks. They provide guidance for situated good scenario based RE practice.

The CREWS approach comprises currently four different method paths. We consider one of these in this paper, namely the one which supports the construction of detailed requirements of both system interactions and system internal assuming that the goal of the system has been agreed upon. The path deals with three types of requirement chunks referred to as *contextual*, *system interaction* and *system internal* RCs

A *contextual requirement chunk* captures a possible manner to fulfil a business goal such as :

> *'Improve services to all bank customers',* or
> *'Reduce the costs of cash provision to normal bank customers of 10%'*.

It couples a *design goal* and a *service scenario.* The design goal expresses one possible manner to fulfil the business goal whereas the service scenario describes the flow of services among agents of the organisation (one being the system itself) which are necessary to fulfil the design goal. For example :

'Improve services to our bank customers by providing cash with ATMs'[2],
'Improve services to our bank customers by using a Web server',
'Improve services to our bank customers by accelerating transactions', and
'Reduce the costs of cash provision to normal bank customers of 10% by charging the cost of cash withdrawals from other banks' customers'

are examples of design goals while :

'The bank customer gets a card from the bank',
'The bank customer gets an account report from the bank web server',
'The ATM transmits the transactions details to the bank consortium', and
'The bank consortium broadcasts the charge costs to the banks of the consortium'

are examples of services in service scenarios. Each of the services are further detailed by system interaction RCs at the system interaction level.

A *system interaction requirement chunk* captures one way to provide a service identified at the previous level. It couples a *service goal* and a *system interaction scenario*. A service goal expresses a way of providing a service identified in a service scenario and therefore, establishes the hierarchical link with a contextual chunk. The associated system interaction scenario describes a flow of interactions between the system and its users which either fulfils the service goal or illustrates the failure of its achievement. Detailing the user/system interactions with the ATM includes exploring the various normal and non-normal behaviours of the ATM and its users in different system interaction scenarios.

A *system internal requirement chunk* details one possible way in which the system may internally perform an interaction identified in a system interaction scenario. It combines a *system goal* and a *system internal scenario*. A system goal such as :

'Verify the card validity following the protocol of the Euro-Bank consortium'

expresses a manner to perform an action which has been identified in a system interaction scenario ; in our example :

'The ATM verifies the card validity'.

The associated system internal scenario describes the flow of interactions among the system objects to fulfil the system goal.

The method path aims at supporting the specification of system requirements by refining a given design goal. The path is illustrated in figure 1. The square boxes represent steps in the process which are supported by method chunks. The arrows represent the flows that can be followed. Starting with one given design goal, it is possible to investigate alternative goals (flow1). Following flow 2, scenarios for each alternative may be authored and therefore, at this level, several alternative architectures of services can be envisioned and evaluated. Once one alternative has been selected, the corresponding contextual RC is refined following flow 3. The service scenario helps discover goals of service of the designed system. Each of these goals is illustrated by scenarios (flow 4) describing interactions between the designed system and its users. During the authoring of system interaction scenarios, the verification phase (flow5) is particularly important because each identified interaction will serve to discover a goal of function of the system. This is done at the system internal level (flow6) where interaction goals are discovered from the system interaction scenarios, and are operationalised by system

[2] ATM is the acronym of Automated Teller Machine

internal scenarios (flow7). The system internal scenarios explore the operations that the objects of the system have to perform to fulfil an interaction. The set of system interaction scenarios and of system internal scenarios can then be used to design the system properly.

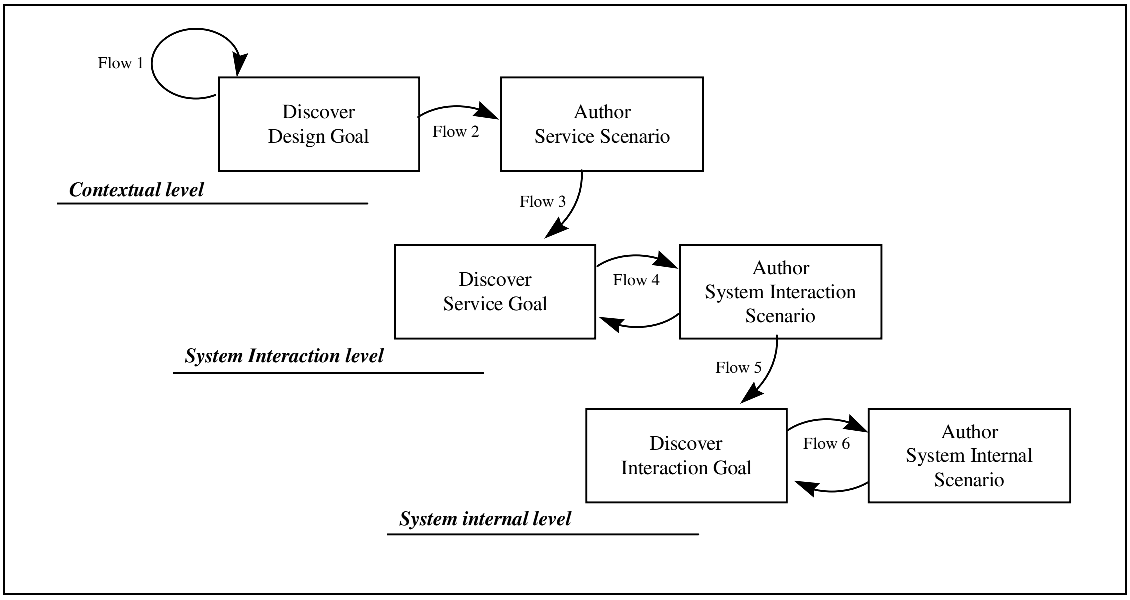

Figure 1 : Example of application of the CREWS process.

We have illustrated one of the CREWS method path for modelling goals and producing scenarios. Numerous goals and scenarios can be produced at each different level of abstraction. The following section addresses the issue of their organisation. First, it deals with the CREWS experience and lessons learned from application of the process. Then, the solution for organising scenarios collections is presented.

3. Structuring a collection of scenarios

3.1 Lessons learned and key points

The lessons learned from our experience of the CREWS approach at its different stages of development lead us to the three following main conclusions.

(1) Need for an external factor to support scenario characterisation.

The structuring of a collection of scenarios requires to define a discriminating and classifying factor. It seems to be difficult to find this factor looking into the internal contents of a scenario Instead it is possible to take an external position. The one we chose is a usage perspective asking the question : 'what is the scenario for ?'. Our answer is that a scenario aims to illustrate a goal. Thus, our proposal is to associate a goal to every scenario and to use goals as a structuring mechanism. The CREWS approach is based on a tight coupling of a scenario and a goal into a requirement chunk. Since a goal is intentional and a goal operational in nature, a requirement chunk is a possible way in which the goal can be achieved. However, the coupling can be looser and the goal associated to a scenario looked upon as an external and descriptive property. Our position is quite convergent with the ones of a number of authors. Coupling goals to scenarios is quite a reoccurring guideline of good practice [3,11,12]. We believe that the coupling can help structuring scenarios collections.

(2) Need for an abstraction mechanism.

The entire system design community relies on abstraction. Evidently scenario based approaches must integrate abstraction mechanisms. The question is 'which ones ?'. One can be tempted to use the goal again. In goal modelling, goals are organised in hierarchies. These goals are expressed at different levels of detail, goals associated to the hierarchy leaves being more detailed than the ones of the hierarchy root. However the levels result from the use of the decomposition mechanism, by which a goal is decomposed into a set of goals connected trough ANDs and ORs. .There is no clear use of an abstraction mechanism but instead abstraction is a derived product of decomposition. The other possibility is to base abstraction on the scenario itself. Some practices in the CREWS steering committee seem to follow that way. In our proposal, abstraction is directed by the scenario part of the requirement chunk. We view every interaction at a level i detailed by an entire scenario at level i+1. More precisely, the proposed abstraction mechanism operates on the two parts of the pair <goal, scenario> and every interaction in a scenario at level i is looked upon as a goal to be achieved at level i+1.Abstraction establishes a vertical relationships among scenarios which is distinct from the horizontal relationships introduced below. We refer to this relationship as the refinement relationship.

(3) Need for composition and alternative mechanisms.

Clearly the AND/OR relationships used in goal modelling to organise goal hierarchies can be extended to scenarios through their coupling with goals. Examples of such AND / OR hierarchies can be found, in the NATURE process theory [5], in KAOS [4], or in F3 [2] among others. However, in the usual goal decomposition strategies, the goal *decomposition* (AND) and the goal *refinement* (OR) involve at the same time vertical and horizontal exploration. Indeed, in these approaches, the components (respectively the alternatives) of a goal have at the same time the property of being less abstract than the higher goal, and complementary (respectively alternative). In our view, the AND and OR relationships should be distinguished from the vertical relationship of refinement, and therefore be purely horizontal.

To sum up, our proposal to structure collections of scenarios is based on three types of relationships, namely the *refinement relationship*, the *OR relationship* and the *AND relationship*. These three relationships are presented in the following with examples taken from the CREWS approach.

3.2 Refinement relationships

Refinement relationships relate scenarios at different levels of detail. These relationships are set between scenarios through a border between two adjacent levels of abstraction. Refinement is used to deploy a part of a scenario at level *i* into scenarios at level *i+1*. The number of levels can be either predefined or free. In the latter case the stakeholder will decide according to the situations he/she is faced to whereas in the former case the method prescribes the levels through which it recommends to elicit scenarios.

The figure 2 illustrates refinement as prescribed in the CREWS method chunk considered in this paper. As introduced in section 2, there are three predefined levels of abstractions, which are, in increasing order of detail, the *contextual level*, the *system interaction level*, and the *system internal level*. In the example presented in figure 3, the refinement relationships are set between requirement chunks:

- *from* a contextual RC
  to a system interaction RC, and
- *from* a system interaction RC
  to a system internal RC.

For example, the system interaction requirement chunk *'Withdraw cash from ATM in a normal way'* refines the contextual requirement chunk *'Improve services to our bank customers by providing cash with ATM'*. Indeed, an interaction of the service scenario attached to the latter is detailed by the system interaction scenario attached to the former.

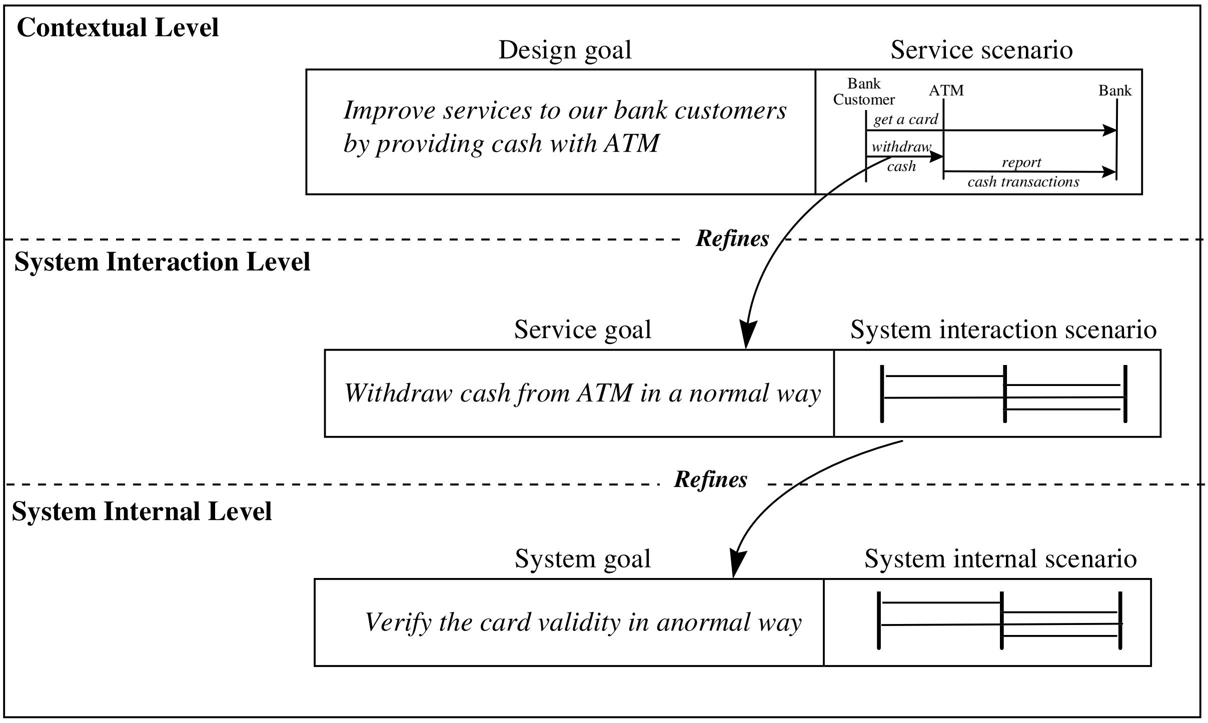

Figure 2 : Example of refinement between three levels of abstraction.

The CREWS method chunk supports abstraction with refinement rules [15]. One of these rules consists of viewing an interaction at level i as a scenario at level i+1. As an example, let us considered the service scenario *'Improve services to our bank customers by providing cash with ATM'*. One of the interactions included in this scenario, *'The bank customer withdraws cash from the ATM'* is refined at the service level by the RC *'Withdraw cash from ATM in a normal way'*. Similarly, the RC *'Verify the card validity in a normal way'* refines the interaction *'The ATM verifies the card validity'* of the scenario of the system interaction RC. Therefore, the two requirement chunks are related by a refinement relationship.

3.3 AND relationships

The AND relationships are set between *complementary scenarios*, in other terms scenarios that need each other to cover more «completely» the description of the designed system. Contrarily to the refinement relationship, the AND relationship is commutative : the combination of the scenarios 'Sc1 AND Sc2' is equivalent to 'Sc2 AND Sc1'. Besides, in opposition to the refinement relationship, the AND relationship can only relate scenarios that are at the same level of abstraction. Finally, the AND relationship is transitive. This means that if a scenario

Sc1 is ANDed[3] to a scenario Sc2, and Sc2 ANDed to the scenario Sc3, then Sc1 and Sc3 are complementary.

In the CREWS approach, AND relationships relate requirement chunks. The goals of ANDed requirement chunks refine the same high level goal. For example, in figure 3, the requirement chunk associated to the goal :

'Withdraw cash from ATM',

is complementary to the requirement chunk associated to the goal :

'Report transactions to the bank',

as both are related with AND relationships to the requirement chunk with the goal :

'Get a card from the bank'.

These goals all participate to the same design goal :

'Improve services to our bank customers by providing cash with an ATM',

that they all refine.

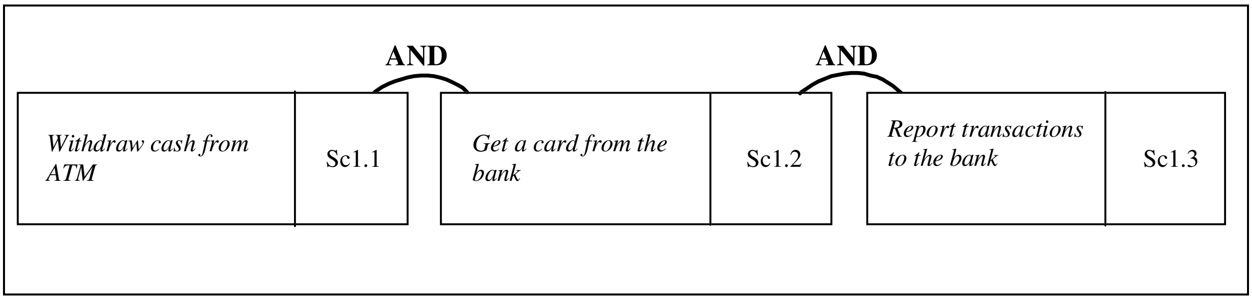

Figure 3 : Example of AND relationship.

Complementary requirement chunks can be found at any level of abstraction. In the CREWS approach, AND relationships are set between complementary contextual requirement chunks, between system interaction requirement chunks, and between system internal requirement chunks. For example, at the system internal level, one can use AND relationships to express that the system internal requirement chunks associated to the system internal goals :

'Verify the card validity'
'Eject the card'
'Print a receipt'
'Deliver cash'
'Verify the code validity'
'Verify the amount validity'
'Get connection to the bank'

are complementary.

Combining a refinement relationship to an AND relationship between requirement chunks is equivalent to goal decomposition in usual goal modelling techniques. However, these techniques do not distinguish the change of level of abstraction from the complementarity of goals. In order to express the complementarity between two goals, it is necessary to consider the higher level goal, and in order to decompose a goal, it is necessary to look at all the

---

[3] We say of requirement chunks which are related by an AND relationships that they are 'ANDed'

complementary goals (looking at only one is not allowed in goal decomposition). In the proposed approach, distinguishing between AND relationships and refinement relationship gives more freedom in reasoning. It allows us on one hand, to express partial refinements (those which are not completed but can be completed later on) and on the other hand to express that RCs are complementary. Thus, the proposed approach is more modular as reasoning on AND relationships and on refinement relationships for the same goal set can be disconnected in time.

From the scenario perspective, the AND relationship between scenarios is similar to the 'extends' relationship between use cases of the Jacobson's approach. The ‘extends’ link expresses that scenarios (respectively use cases) supplement one another [6]. However, the 'extends' relationship does not allow to distinguish within 'supplementary' use cases those that provide supplementary information by considering things in more detail from those that bring complementary knowledge about the designed system. This distinction is made in our approach by differentiating abstraction from complementarity.

Finally, let us notice that the AND relationship allows to complete requirement chunks, but does not ensure the property of completeness. In the example of figure 3, one can still find other complementary requirement chunks, such as :

‘Restore cards to the bank customers’,
‘Fill in the ATM with paper’, and
‘Fill in the ATM with cash’.

3.4 OR relationships

Scenarios related through OR relationships, represent alternative ways to fulfil the same goal. Like the AND relationship, the OR relationship is horizontal. Besides, the OR relationship is commutative and transitive.

In the CREWS approach, the OR relationship is expressed between two requirement chunks at one level of abstraction. We propose to distinguish two types of OR relationships*: tactical relationships* and *strategic relationships*

Tactical relationships associate two requirement chunks that refine the same high level goal, but in different ways. In figure 4 for example, all scenarios Sci correspond to different ways of achieving .the service goal ‘withdraw cash from......’. The first one corresponds to the normal case whereas the other ones correspond to either variations of the normal case or exceptional cases.

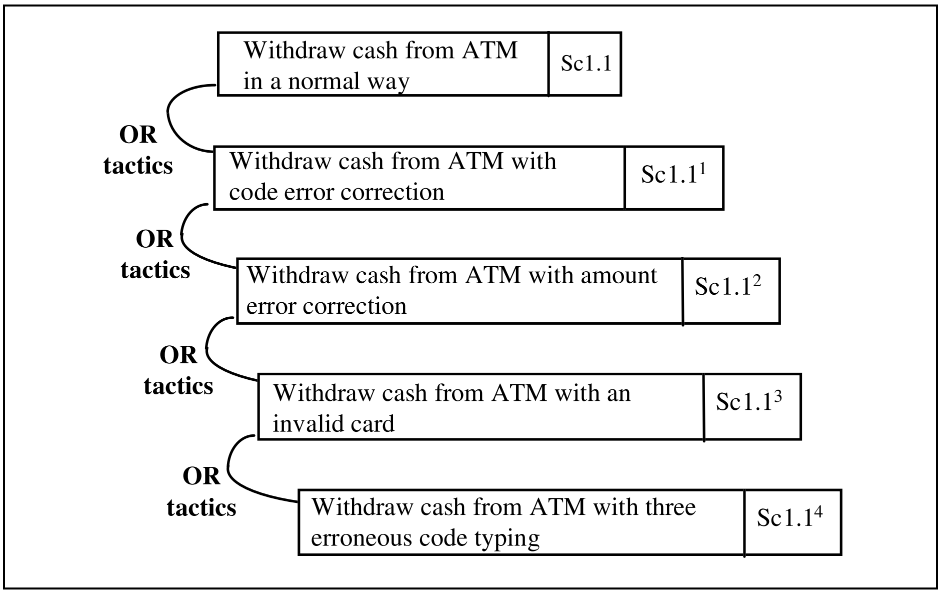

Figure 4 : Example of OR relationships set between alternative tactics.

The use of the ATM with one error in the code typing ($Sc1.1^1$) corresponds to a variation of the normal case (Sc1.1) of cash withdrawal. But the use of the ATM with three errors in the code typing ($Sc1.1^4$) corresponds to an exceptional behaviour. It is exceptional in the sense that it leads to a failure in the achievement of the 'cash withdrawal' goal.

Strategic relationships correspond alternative goals of a given high level goal. Figure 5 gives an example of such strategic relationships for the ATM example. The three following goals :

> '*Improve services to our bank customers by providing cash with ATMs',*
> *'Improve services to our bank customers by providing account management facilities with ATMs',* and
> *'Improve services to our bank customers by accelerating transactions',*

are alternative design goals of the business goal *'Improve services to our bank customers'*. The services of the ATM and of the bank will be different depending of the choice to design a normal ATM, an ATM with transaction management facilities or an ATM supporting accelerated transactions. The three options represent strategic options among which only one will be selected

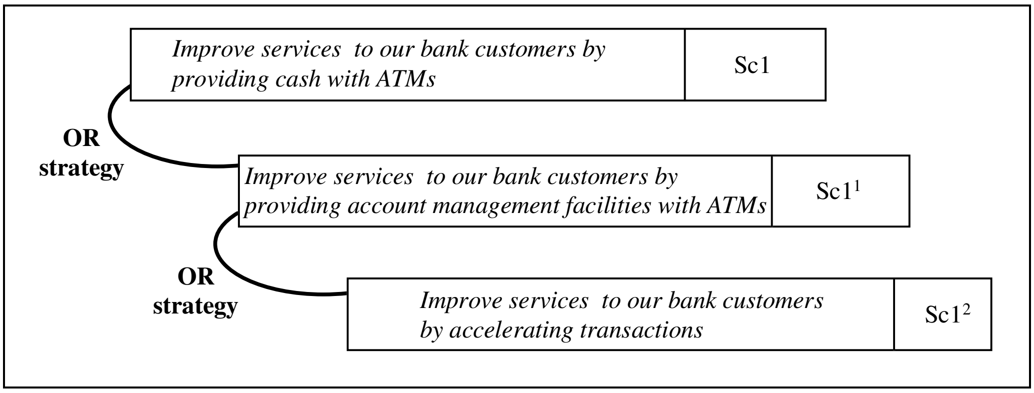

Figure 5 : Example of OR relationship relating alternative strategies.

Combining OR and refinement relationships is equivalent to the goal decomposition technique used in goal modelling approaches Seen from the scenario perspective, the OR relationship can be compared to the Jacobson's 'uses' association. Indeed, the 'uses' association aims at avoiding redundancy by factorising common sub-flows of use cases, and allows to differentiate between alternative sub-flows. The ‘uses’ association is syntactical whereas the OR relationship as proposed here is semantic. The OR relationship is comparable to what Cockburn calls ‘variation’. However distinguishing strategic from tactic variations allows to differentiate alternative scenarios for a single choice of design and alternative scenarios representing different possibilities of design.

In this section, we have presented, defined and illustrated the three types of relationships that we propose to organise collections of scenarios namely, *refinement relationships*, *OR relationships* and *AND relationships*. The three relationships are based on the assumption that every scenario is characterised by the goal it aims to illustrate. AND and OR relationships are inherited from goal decomposition techniques. But they are horizontal relationships disconnected from the vertical, abstraction based relationships. This de-coupling facilitates a modular reasoning on the two kinds of scenarios connections : abstraction can be used separately from composition and alternative. This should ease the process of structuring large collections of scenarios. As experienced with the CREWS approach it makes also easier the definition of process guidelines.

In the following section, we shall propose heuristic rules of good practice for building 'quality' networks of requirement chunks.

4. Quality management

Relating requirement chunks with AND, OR, and refinement relationships allows us to build complex networks including multiple design perspectives on complex behaviours of agents, at different levels of abstraction. Multiple refinement relationships are allowed between two abstraction levels, and at the same level of abstraction, complex combinations of ANDs and ORs are also permitted. In order to improve the management of the scenario network, we propose a set of *quality properties* to be respected and *heuristic guidelines* to guide the construction of a network of scenarios.

The quality properties that we propose are defined as constraints on AND / OR / and refinement relationships. These are :

- Only one refinement relationship should be set between two abstraction levels i and i+1.
- A RC should be ANDed to at most two requirement chunks.
- A RC should be ORed to at most two requirement chunks.
- A RC ANDed to another RC should be ORed to at most one requirement chunk.
- There is no cycle in a chain of AND relationships.
- There is no cycle in a chain of OR relationships.

Let us call *root* the father requirement chunk in a father/sons abstraction hierarchy. Then, the new quality property :

- At any abstraction level, the root should be ANDed to at most one requirement chunk.

Can be added to the others ;

Given a chain of ANDed (ORed) requirement chunks, let’s call the *ANDleaf* (respectively the *ORleaf*) the requirement chunk which is a leaf of the hierarchy. We call *ORroot* a requirement chunk which belongs both to a chain of ANDed requirement chunks and to a chain of ORed requirement chunks. The list of quality properties can be completed as follows :

- Any ORroot should have at most one alternative tactic.
- Any ORroot should have at most one alternative strategy.
- Therefore, any ORroot should be ORed to at most two requirement chunks.

The quality properties listed above allow to organise networks of RCs in a similar way that the one presented in figure 6. In this example, the requirement chunk RC1 is at the same time the root, ANDleaf and ORleaf of the level 1. At the level 2, the root is RC1.1. The associated ORleaf is then $RC1.1^2$, and the ANDleaf is RC1.3.

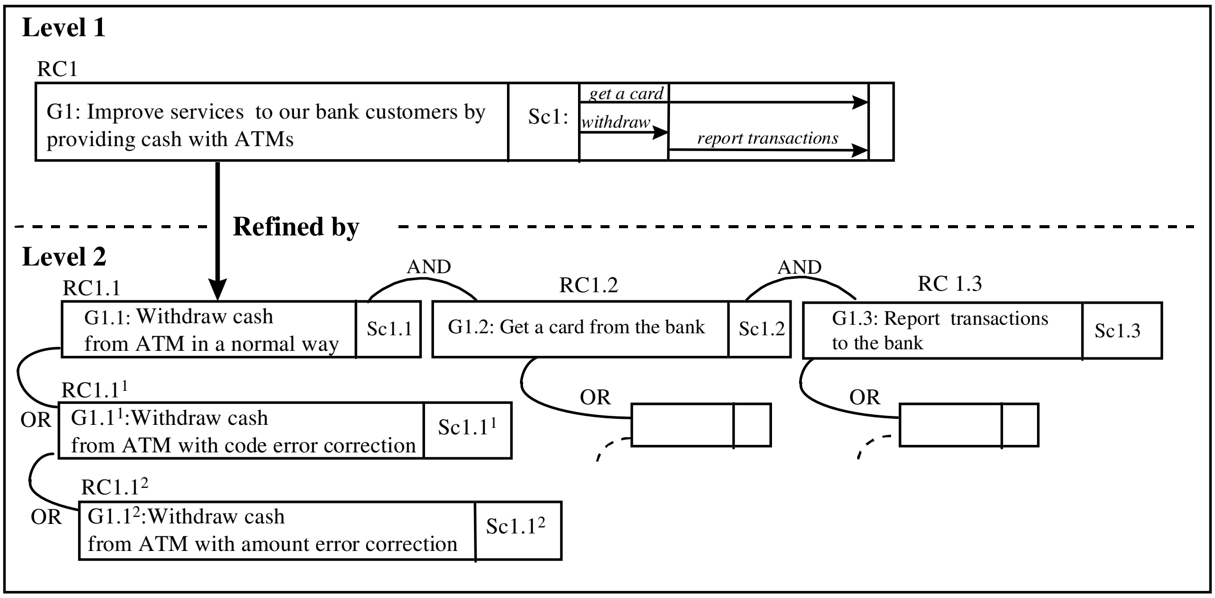

Figure 6 : Example of a scenarios network.

Complementarily, in order to control that a network is built with respect to the proposed quality properties, we defined heuristics guidelines. Each guideline is situated : it corresponds to possible case of integration of a new RC into the network. Guidelines are presented with the following template :

- the intention to be achieved, presented in italic,
- steps to be followed and corresponding actions

Integration Rule INT1 :

To AND a new requirement chunk RC1 to an existing requirement chunk RC2 :

- identify the ORroot RC3 of the chain of ORed requirement chunks RC2 belongs to,
- identify the ANDleaf RC4 of the chain of ANDed requirement chunks RC3 belongs to,
- create an AND relationship between RC1 and RC4 ; RC1 is now the ANDleaf of the chain.

Let us illustrate this rule by relating a requirement chunk RC to the requirement chunk $RC1.1^1$. The ORroot associated to this requirement chunk is RC1.1 : *'Withdraw cash from ATM in a normal way'*. Then, the ANDleaf associated to RC1.1 is the requirement chunk RC1.3 *'Report transactions to the bank'*. Applying the rule leads to completing the sets of requirement chunks with RC, and relate RC1.3 to RC with an AND relationship. The result, is illustrated in figure 7.

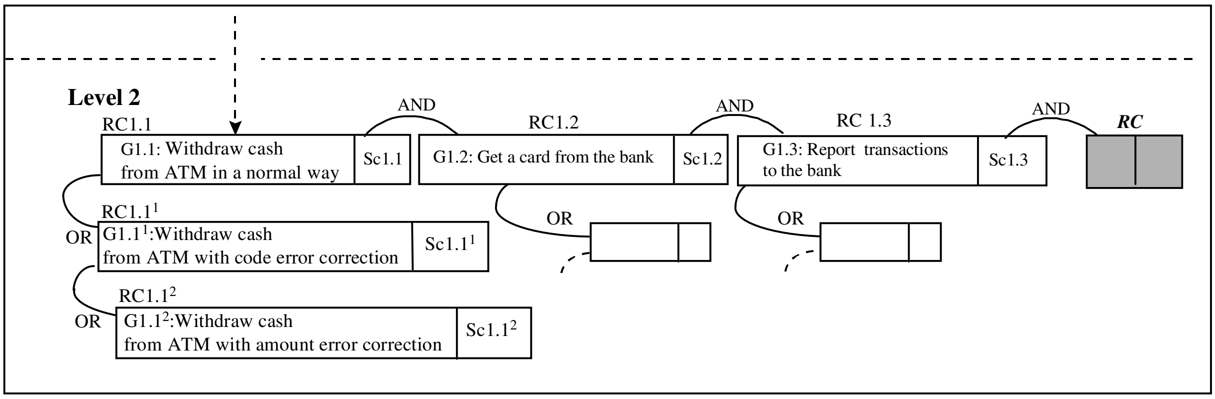

Figure 7 : Example of application of the rule INT1.

Integration Rule INT2 :

To OR a new requirement chunk RC1 to an existing requirement chunk RC2 :

- identify the ORleaf RC3 of the chain of ORed requirement chunks RC2 belongs to,
- create an OR relationship between RC3 and RC1 ; RC1 is now the ORleaf in the chain.

Let's illustrate this rule starting again from the situation of the network presented in figure 6. To OR a requirement chunk RC to the requirement chunk $RC1.1^1$, the guideline proposes to search for the ORleaf related to $RC1.1^1$. This requirement chunk is the one situated at the end of the chain of ORed requirement chunks : it is $RC1.1^2$. Then, RC has to be added to the network, and related to $RC1.1^2$ by an OR relationship. The result of the integration is illustrated in figure 8, the new requirement chunk is in grey.

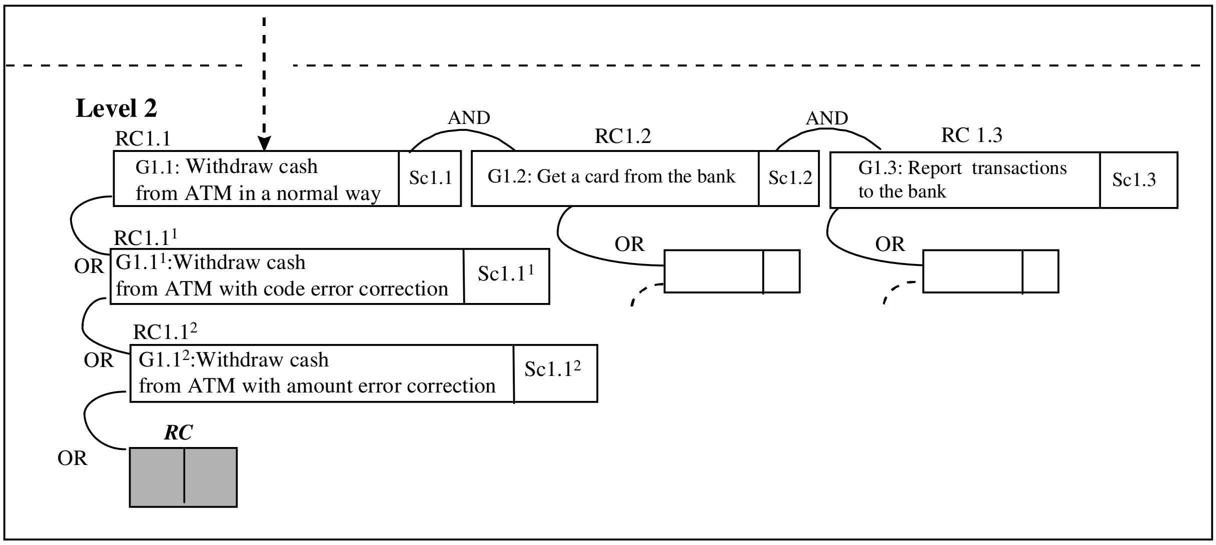

Figure 8 : Example of application of the rule INT2 of integration.

Let us note that ORing RC to RC1.1, or to $RC1.1^2$ would have given the same result. On the contrary, ORing RC to any of the requirements chunks in the chains of which RC1.2 and RC1.3 are the ORhead would have led to a different result.

Integration Rule INT3 :

To create a refinement relationship between a new requirement chunk RC1 at level i+1 and a requirement chunk RC2 existing at level i :

- if there is already a refinement relationship between levels i and i+1, then

- identify if RC1 should be ANDed or ORed to the root of the level i+1,
- follow the guidelines INT1 and INT2 to relate RC1 and the root of level i+1,
- if there is no refinement relationship between levels i and i+1, then
  - create a refinement relationship from RC2 to RC1.

Two cases of refinement relationship creation are considered in this rule. In the first case, a refinement already exists between the two levels at which the new refinement is to be created. In the second case there is no refinement already existing.

Refining any of the requirement chunks of level 2 in figure 6 by a new requirement chunk RC corresponds to the second case. In this case, RC is integrated in the network, and a refinement relationship is set between it and the refined requirement chunk. This is illustrated in figure 9 by the refinement of RC1.3.

On the contrary, refining the requirement chunk RC1 in figure 6 corresponds to the first case. Indeed, RC1 is already refined. Then, the integration rule proposes to reason about the relationship between the new requirement chunk and the other requirement chunks at the level at which it is to be integrated. Depending on whether it is complementary or alternative to another requirement chunk, the rule INT1 or INT2 can be applied.

The integration of strategic alternative is similar to the integration of tactic alternatives. It is not illustrated in this paper for the sake of place and clarity.

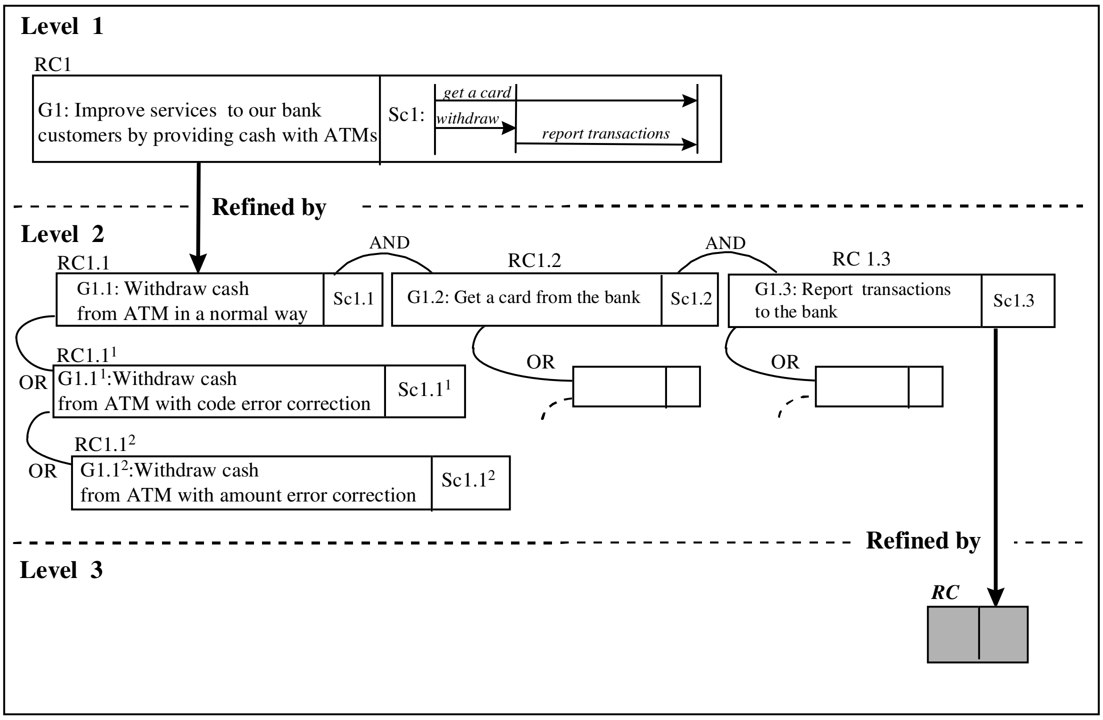

Figure 9 : Example of application of the rule INT3 of integration.

The three integration rules presented above are implemented in the CREWS tool *L'ÉCRITOIRE*. Indeed, this tool implements the goal discovery and scenario authoring steps presented in section 2. At each discovery of a new goal, a requirement chunk is created and inserted in the network of requirement chunks. This network is recorded as a database of requirement chunks and AND / OR / Refinement relationships. The choice of the relevant integration rule depends on the strategy of discovery which is applied.

5. Conclusion

Managing a high number of scenarios is seen today as a crucial issue of scenario based requirements engineering. The solution proposed in this papers consists in organising scenarios into networks. Such a network relates scenarios to goals, and relates each other through AND, OR, and refinement relationships. This solution is generic in the sense that any kind of scenario can be organised this way. In addition it is compatible with AND/OR goal trees. Combining goals to scenarios is positive for the elicitation of both. The solution is illustrated in this paper with the CREWS approach in which the concept of requirement chunk couples tightly goals to scenarios. However the relationship between both can be weaker.

In order to measure its scalability, the formalism has been tested with the real world example of the business process re-engineering of a European electrical company [10].

The AND / OR / Refinement network has been implemented in the CREWS tool *L'ÉCRITOIRE*. with guidance rules for the construction of well formed networks. Open issues concerning the construction of such networks include the need of negotiation techniques to limit the number of generated scenarios, the need of relationships with the other artefacts used in the RE process [9] and the need of viewers and editors.

References

[1] C. Ben Achour, *Guiding scenario authoring.* Submitted to the Euro-Japanese conference, 1998.

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

[3] A. Cockburn, *Structuring use cases with goals*. Technical report. Human and Technology, 7691 Dell Rd, Salt Lake City, UT 84121, HaT.TR.95.1, available at http://members.aol.com/ acockburn/papers/usecases.htm, 1995.

[4] A. Dardenne, A. van Lamsweerde, S. Fickas, *Goal directed requirements acquisition.* Science of Computer Programming, 20(1-2), pp.3-50, April 1993.

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

[6] I. Jacobson, M. Ericsson and A. Jacobson, *''The object advantage, business process reengineering with object technology''.* Addison-Wesley Publishing Company, 1995.

[7] M. Jarke, Dagtsuhl Workshop : Proceedings. To appear.

[8] M. Jarke, K. Pohl, P. Haumer, K. Weidenhaupt, E. Dubois, P. Heymans, C. Rolland, C. BenAchour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, S. Minocha, *Scenario use in European software organisations – Results from site visits and questionnaires.* CREWS report 97-10, available at http ://SUNSITE.informatik.rwth-aachen.de/CREWS/reports.htm, 1997.

[9]. 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*. In Third IEEE International Symposium On Requirements Engineering RE'97, Antapolis, Maryland, IEEE Computer Society Press, pp. 44-53, 1997.

[10] S. Nurcan, G. Grosz, C. Souveyet, *Describing Business Processes with a Guided Use Case Approach.* Proceeding of the CAiSE'98 Conference on Advanced Information Systems Engineering, to appear in June 1998.

[11] C. Potts, *Using schematic scenarios to understand user needs.* Proc. DIS'95 : Designing Interactive Systems, Ann Arbor, MI, pp247-256, August 23-25 ,1995.

[12] C. Potts, *Fitness for use : the system quality that matters most*. Proceedings of the 3rd International Workshop on Requirements Engineering : Foundation for software quality, REFSQ'98, Barcelona, Spain, pp. 15-27, 16-17 June, 1997.

[13] C. Rolland, C. Ben Achour, *Guiding the construction of textual use case specifications.* To appear in Data and Knowledge Engineering Journal, 1998.

[14] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, P. Heymans, *A Proposal for a Scenario Classification Framework*. Requirements Engineering Journal, 3 :1, 1998.

[15] C. Rolland, C. Souveyet, C. Ben Achour, *Guiding goal modelling using scenarios.* Submitted to TSE special issue on Scenario Management, 1998.