CREWS Report Series 98 - 12

GUIDING SCENARIO AUTHORING

C. Ben Achour

Université Paris 1 - Sorbonne
17 rue de Tolbiac. 75013 Paris - France

Université Paris 6 - Pierre et Marie Curie
4, place Jussieu. 75005 Paris - France

Appeared in the proceedings of the Eight European - Japanese Conference on Information Modelling and Knowledge Bases. May 25-29, Vamala, Finland.

Guiding Scenario Authoring[1]

Camille Ben Achour
Université Paris 1 - Sorbonne
17 rue de Tolbiac. 75013 Paris - France
Université Paris 6 - Pierre et Marie Curie
4, place Jussieu. 75005 Paris - France
camille@univ-paris1.fr

Abstract : *Since a few years, scenario based requirements engineering approaches have gained in popularity. Textual scenarios are narrative descriptions of flows of actions between agents. They are often proposed to elicit, validate or document requirements. The CREWS experience has shown that the advantage of scenarios is their easiness of use, and that their disadvantage stands in the lack of guidelines for authoring. In this article, we propose guidance for the authoring of scenarios. The guided scenario authoring process is divided into two main stages : the writing of scenarios, and the correcting of scenarios. To guide the writing of scenarios, we provide style and contents guidelines referring to a conceptual and a linguistic model of scenarios. To guide the correcting of scenarios, we propose a set of enactable rules. These rules aim at the clarification, completion and conceptualisation of scenarios, and help the scenario author to improve his scenarios until acceptable quality in the terms of the former scenario models. The paper presents both style / contents guidelines, and correction rules, and illustrates them with the London Ambulance Service example.*

Keywords : Scenario, Scenario Authoring, Scenario Correction, Requirements Engineering.

1. Introduction

Scenarios have recently gained attention in the field of Requirements Engineering (RE). Many approaches have been proposed until now. For example Kyng proposes to use scenarios to envision the requirements [11], Potts to elicit the requirements [15], Jacobson to analyse the objects of the application domain [9], etc. Scenarios may have different forms, contents, coverage, and purposes [19]. However, our investigations show that in the literature, most of the approaches lie on textual scenarios [19], and that in the practice, in about 80% of scenario-based projects, scenarios are described under the textual form [27]. The growing number of practitioners demanding for more ‘informality’ in the requirements engineering process [12] seems to confirm this trend. However, as show our enquiries at several major European companies, requirement engineers still need help in the process of authoring scenarios [10] [3].

In this paper, we are interested in the guidance of the so called ‘scenario authoring’ process [20]. By scenario authoring, we mean to write, clarify, complete and conceptualise scenarios. In addition, we focus on *textual scenarios*, that is to say descriptions of stories of system usage in (more or less structured) natural language. In the literature, textual scenarios are also referred to as *stories* [6], *cases* in *use case descriptions* [18] or *scripts* [22].

[1] The work presented in this paper is funded by the European Community within the framework of the ESPRIT LTR project CREWS (*Cooperative Requirements Engineering With Scenarios*) n°21903.

Authoring is part of a two-step process of <scenario authoring - goal discovery>. Indeed, the CREWS experience in scenario based requirements engineering shows that goals and scenarios are complementary [14,21]. On the one side, scenarios are used to discover new goals, and on the other side, goals are the input for scenario authoring. In order to guide the scenario author (SA) in the authoring step, we provide writing guidelines and verification rules. The writing guidelines aim at conducting the SA towards a quality writing of textual scenarios. The quality of textual scenarios refers to a conceptual and a linguistic model of scenarios. While he / she writes a scenario, the SA can evaluate its quality by applying enactable rules. These rules guide the SA in structuring, clarifying and completing scenarios.

The next section describes the conceptual model of scenarios referred to along this paper. Section 3 tackles the linguistic aspect of textual scenarios. Section 4 overviews the process of scenario authoring. Sections 5 and 6 provide respectively the guidelines for writing textual scenarios and the rules for correcting them. The last section discusses the results presented in this paper. Illustrations taken from the London Ambulance Service (LAS) example [8] are given all along this paper.

2. The conceptual model of scenarios

A scenario expresses an example of behaviour of a system. Thus, a scenario is *« one possible behaviour limited to a set of purposeful interactions taking place among several agents »* [4]. On the contrary, use cases describe several possible uses of a system. Therefore, a use case is composed of several scenarios. We say how to integrate scenarios into a use case in [20]. The Figure 1 presents the model of scenarios with the OMT notation [23].

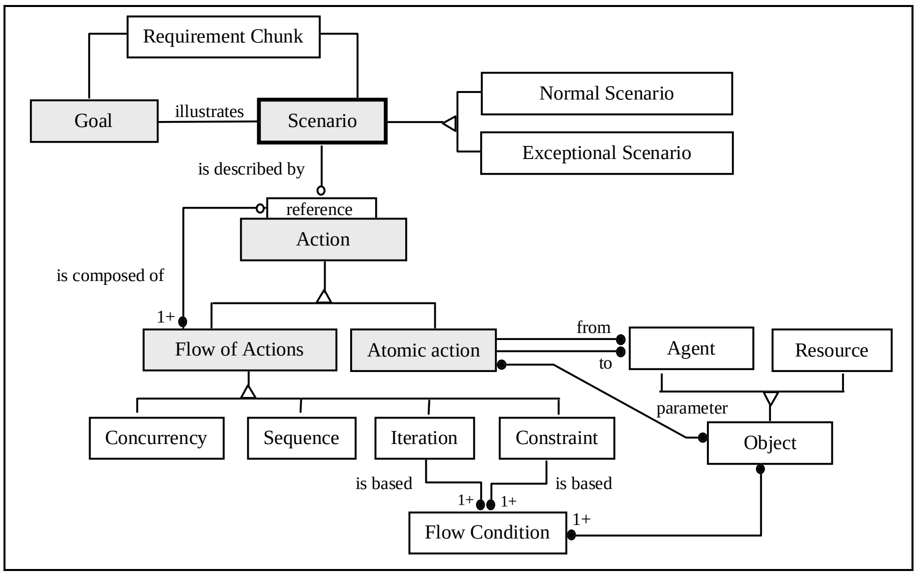

Figure 1 : The scenario model.

The CREWS approach defends a tight coupling of goals and scenarios into *requirement chunks* [16, 21]. To any scenario is attached a *goal*. Goals express in part the requirements for the designed system. They are written as a verb together with one or several parameters, and they are illustrated by scenarios. More information can be found on how to write goals in [17], and [20].

Two kinds of scenarios are distinguished : normal and exception scenarios. A *normal scenario* illustrates a way to fulfil a goal. On the contrary, an *exceptional scenario* illustrates a case of failure in the goals fulfilment. In the following, we call a scenario by the name of the goal it illustrates. For example whereas the scenario *'Allocate incidents to an ambulance crew* *in a normal way'* corresponds to a normal scenario of incident allocation to one of the different ambulance crews in the London Ambulance Service, the *'Allocate incidents to an ambulance crew with no available ambulance in the LAS'* scenario describes an exception in which the incident allocation goal is not achieved. It must be noted that a scenario is not called exceptional because it is infrequent. For example, the exception scenario *'Allocate incidents to an ambulance crew for a duplicate call'* is very frequent (as shown in [8]), however it is an exception scenario.

The scenario model shows that a scenario is composed of one or several *actions* (the multiplicity appears by the black dot together with the sign '+1'). Actions are of two types : atomic actions and flows of actions. To any action is attached a *reference* which identifies it in a unique way. Action references are thus used in any action composition : between a scenario and an action, and between a complex flow of actions and other actions.

An a*tomic action* is going '*from'* one and only one agent '*to'* one and only one agent, and affects some '*parameter*' '*object'* (in the OMT notation unicity is represented by the absence of dot, multiplicity by black dots). The agents may be the designed system, its users, its component objects, etc, and the resources the passive objects that they manipulate. Both may be involved into several actions. Moreover, in one atomic action, the 'from' and 'to' agents can be the same. For example, *'The CAD system checks if the call is not in duplicate'* is an example of internal action, in which the 'from' and 'to' agents are the same : the Computer Aided Despatch system. On the contrary, *'The CAD system sends the mobilisation order to the relevant ambulance crew*' is an atomic action of communication. Its agents are 'the system' and 'the relevant ambulance crew', and its parameter is the resource 'the mobilisation order'.

The composition of actions is expressed in *flows of actions* which are refined into *sequence, concurrency, iteration* and *constraint*, that refer to the connectors of strict sequence, co-beginning parallelism, loop, and condition. For example, the sentence *'The CAD system locates the incident place, and then decides which division is going to deal with the incident'* expresses a sequence between two atomic actions. The sentence '*While it sends the mobilisation decision to the relevant ambulance crew the CAD system updates the availability database'* illustrates concurrency; there is no predefined order between the two concurrent actions. As defined in Figure 1, flows of actions can be themselves composed of other flows of actions. Thus, a sequence of actions can be iterated, concurrent actions constrained, several constraints embedded, etc.

Contrarily to the concurrency and sequence flows of actions, the iteration and constraints are 'based on' flow conditions. *Flow conditions* express the assumptions for which a meaningful behaviour of the scenario agents is defined. Within the potentially infinite set of possible cases that the designed system and its users can face, flow conditions allow the author to express one course of actions in a scenario. In the sentence *'If the call is not in duplicate, then the CAD system locates the incident place'*, the flow condition '*If the call is not in duplicate'* identifies a case of CAD usage. The cases of usage of the CAD system with a call already received for the same incident should thus be described in a separate scenario.

In this paper we use either the unstructured or the semi-structured textual form to represent scenarios. When written in the unstructured textual form, like in Figure 2 (a), a scenario looks like a story in natural language. When a scenario is presented in the semi-structured textual form, like in Figure 2 (b), each of its actions is expressed by a clause in a separate line. We shall detail the relationships between textual scenarios and the scenario model in the following section.

(a)	Provide the ambulance service to London patients	A caller sends an urgent call to the central ambulance control. Then, the central ambulance control allocates the incident to an ambulance crew, and the ambulance crew delivers the medical service to the patient.
(b)	Allocate incidents to an ambulance crew in a normal way	1. The details of the call are entered in the CAD system. 2. If the call is not in duplicate, then 3. The CAD system locates the incident place. 4. It decides which division is going to deal with the incident. 5. The resource status and incident details are displayed to the division resource allocator. 6. If an ambulance and a crew are available at the division, then 7. The resource allocator identifies the resource to be mobilised. 8. He must enter the mobilisation decision within two minutes. 9. The system sends rapidly the mobilisation order to the relevant ambulance crew.

Figure 2 : Unstructured and semi-structured textual representations of scenarios.

3. The linguistic baseline

A scenario should be written correctly in reference to the scenario model. Thus, the scenario model provides the structure of scenarios. Scenarios are themselves written in natural language. Consequently, there is a relationship between the text structure and the scenario model structure which is highlighted in Figure 3.

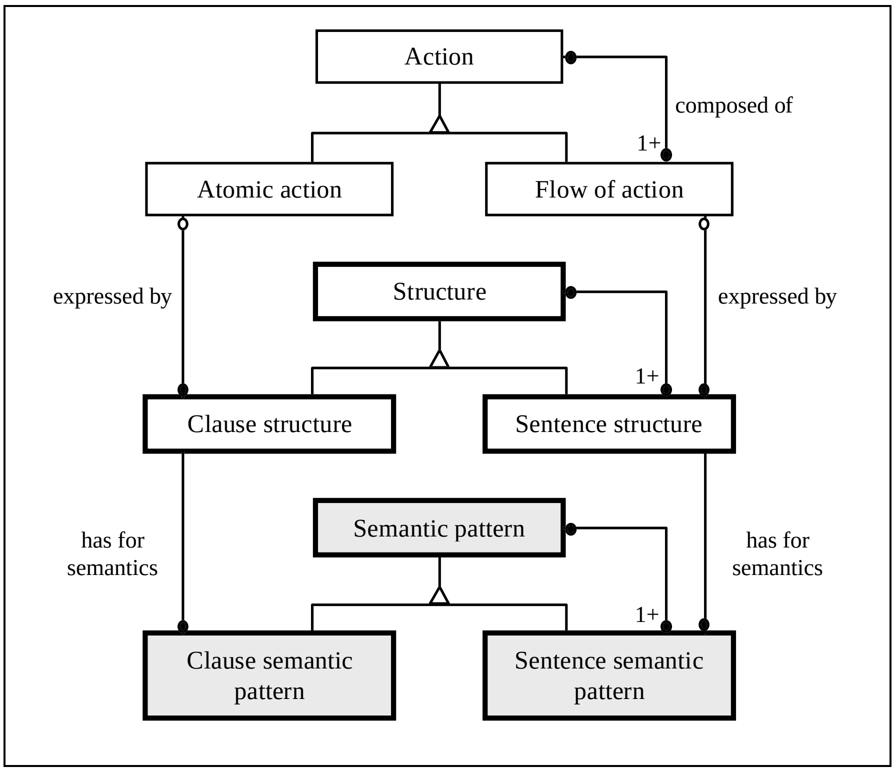

Figure 3 : Relationships between text structure and the action structure.

Scenario actions are refined into atomic actions and flows of actions. Similarly, the text *structures* (see bold boxes in Figure 3) can be decomposed into more elementary structures which are either *clause structures* or *sentence structures*. For example the scenario extract *'The central ambulance control allocates the incident to an ambulance crew, then the ambulance crew delivers the medical service to the patient'* has a sentence structure. It can be decomposed into two elementary clause structures corresponding to : '*The central ambulance control allocates the incident to an ambulance crew*', and '*the ambulance crew delivers the medical service to the patient*'.

Structures correspond to the surface representation of the textual specification. It is also referred to as the syntax. The semantics is provided by *semantic patterns* (see grey light boxes with bold borders in Figure 3). *Sentence* and *clause semantic patterns* reflect respectively the deep aspect of sentence and clause structures. Thus, clause semantic patterns provide the semantics of atomic actions and sentence semantic patterns provide the semantics of flows of actions : the sequence, the concurrency, the repetition and the constraint.

The semantic patterns are patterns of a Case Grammar [7]. They identify a restricted sub set of the natural language semantics which is expected in scenarios. Similarly, the structures identify a restricted syntax that is expected for scenarios. In the following sub sections, we justify the choice of the case grammar, recall its main principles, and propose an adaptation to the expression of scenarios.

3.1 The case grammar

Our approach of scenario authoring uses natural language. As shown in section 2, the scenario model focuses on the notion of action. However, in textual scenarios, actions are expressed informally. Thus, there is a need to fill the gap between the textual representation of actions and their formal counterpart in the scenario model.

As shown in [24], the use of free natural language in RE raises difficult problems. For example ambiguous and implicit statements requiring contextual or domain knowledge are especially dangerous in the framework of requirements engineering where the least misunderstanding may have dreadful consequences. In order to avoid such problems, we restrict the language of scenario expression to : a restricted set of terms, the syntax defined by a finite set of structures, the semantics defined by the scenario model, and the statements that do not require contextual or domain knowledge to be interpreted. We justify the choice of the case grammar to fill the <informal representation - formal model> gap by the fact that it is consistent with these assumptions, and that it focuses on the notion of action.

Fillmore [7] has first shown that, in a clause expressing an action, the semantic relationship between the main verb and its parameters can be systematically characterised. This deep level characterisation called *semantic case* is very different from the surface level grammatical cases of flexional languages such as German or Russian. Many other authors have followed the way opened by Fillmore ([2, 5, 25, 26, 28]), proposing different definitions of the notion of semantic case. A discussion on these various versions of the case grammar can be found in [1, 20].

Our case grammar lies on a finite set of semantic cases such as agent, object, destination, etc. Each semantic case defines the role that an element of a clause plays in relationship with the main verb. A scenario is composed of sentences, each sentence expresses an atomic or a complex action. Atomic actions are expressed as clauses with a main verb and several parameters, where each parameter plays a different role with respect to the verb. For example in the action statement :

‘*(The CAD system)*$_{Agent}$ *(locates)*$_{Verb}$ *(the incident place)*$_{Object}$’

‘locates’ is the main verb, ‘the CAD system’ is the agent of the action, and ‘the incident place’ is the object on which the action applies.

The semantic cases of the case grammar are different from grammatical functions. In the clause *‘the incident place is located’* the grammatical subject (*‘the incident place’*) is the object of the action whereas in *‘The CAD system locates the incident place’*, the subject (*‘the CAD system’*) is the agent of the action. The semantic cases *agent* and *object* are thus different from the grammatical functions subject and direct object. Let us list the semantic cases defined for atomic actions.

Agent : the *agent* (Ag) identifies the entity which undertakes or controls the action. It can be as well a human or a machine, e.g. the designed system itself, an object, or a user. For example, in the action statements :

‘*(The CAD system)$_{Agent}$ (checks)$_{Verb}$ (if the call is not in duplicate)$_{Object}$*’

the ‘CAD system’ is the agent of the action that it both initiates and controls.

Target : the *target* (Tar) designates entities affected by the action. We distinguish two types of targets, objects and results. As opposed to *objects* (Obj), the *results* (Res) are entities affected by the action which do not exist prior to the action, or which exist in abstract form and are made concrete as a result of the action achievement. For example in the action statement :

‘*(The CAD system)$_{Ag}$ identifies (the resource to be mobilised)$_{Res}$*’,

the choice of the CAD system is the result of the action achievement. On the contrary, in :

‘*(The control assistant)$_{Ag}$ enters (the details of the call)$_{Obj}$ (in the CAD system)$_{Dest}$*’,

the target, ‘the details of the call’, is an object because it is supposed to exist before the action is achieved.

Direction : the two types of *direction* (Dir), namely s*ource* (So) and *destination* (Dest) identify respectively the origin and end location of objects to be communicated, such as in :

‘*(The division resource allocator)$_{Ag}$ reads (the resource status)$_{Obj}$ (on the terminal of the CAD)$_{So}$*’, and

‘*(The CAD system)$_{Ag}$ sends (the mobilisation decision)$_{Obj}$ (to ambulance crew)$_{Dest}$*’, respectively.

An important result of the Case Grammar is that verbs which have a similar meaning are used with similar semantic structures. For example, the verbs expressing an action of communication (such as ‘to give’, ‘to take’, ‘to move’, ‘to enter’, etc.) always express the move of an object (information or physical object), initiated by a agent, from a source location to a destination location. Typical semantic structures are defined by patterns of cases called *semantic patterns*. Semantic patterns are represented as templates that permit to associate a semantic case to the different elements of clauses and sentences. In our case, the purpose of the semantic patterns is to define the semantics of the clauses and of the sentences which are respectively expressed in scenarios. Our ontology of semantic patterns presented in Figure 4 is fully directed towards the semantics definition of the different types of actions included in the scenario model.

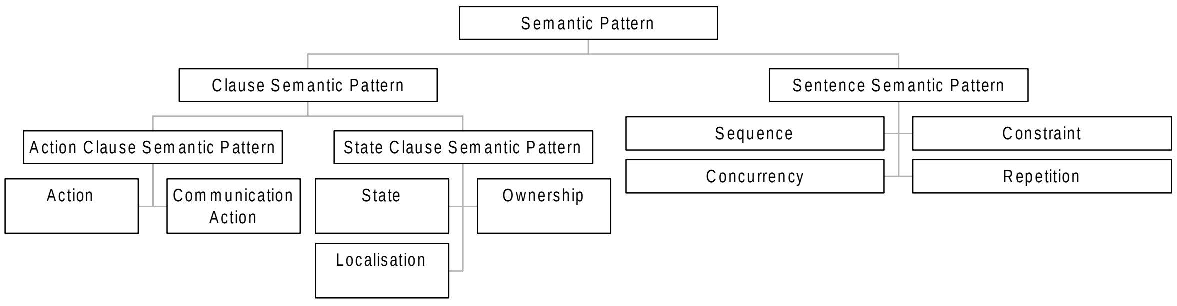

Figure 4 : Ontology of semantic patterns.

This figure shows that our case grammar comprises semantic patterns for sentences and clauses. At the level of clauses, case patterns are *clause semantic patterns* associated to verbs. Clause patterns are related to the intrinsic semantics of verbs, and does neither include any domain knowledge nor contextual knowledge. Clause patterns can define the semantics of actions (e.g. *communication actions*) and static relationships (e.g. *state*, *localisation*, or *ownership*).

State clause semantic patterns provide the semantics of the states of objects. The ownership and localisation define the respective roles in a relationship between two objects, whereas the state pattern identifies the specific static properties of one object. The state clause semantic patterns define the semantics of object states on which can rely the flow conditions of the constraint and iteration flows of actions.

Action clause semantic patterns provide the semantics of the atomic actions of the scenario model by associating a semantic role to the related agent, object and result parameter. They are presented under the form N (V) [C], where N is the name of the pattern qualifying the intrinsic semantics of the verb V, and C is the list of cases to be associated to the elements of the analysed clause. To represent the semantics of a clause in a scenario specification consists in instantiating a clause semantic pattern, that is to say in giving a value to the variables V and C. For example,

Action (V) [Agent ; Object]

being the clause pattern for actions, the instantiated clause pattern :

Action ('check') [Agent : 'the CAD system' ; Object : 'if the call is not in duplicate']

provides the semantics of the atomic action *'the CAD system checks if the call is not in duplicate'* by providing a role to the agents and object parameters of the action.

As illustrated in Figure 4, *sentence semantic patterns* define the semantics of the flows of actions of the scenario specification. Sentence semantic patterns, contrarily to clause semantic patterns define the semantic relationships between two clauses or sentences of a same text. This recursive property of sentence semantic patterns reflects the composition of flows of actions in the scenario model. For example :

Sequence [Before : Action ('identify') [Agent : 'the resource allocator'; Result : 'the resource to be mobilised'] ;
After : Sequence[Before : Communication ('enter') [Agent : 'the resource allocator' ; Object : 'the mobilisation decision' ; Source : 'the resource allocator ; Destination : 'the CAD system'];
After : Communication ('send')[Agent : 'the CAD system' ; Object : 'the mobilisation order' ; Source : 'the CAD system'; Destination : 'the relevant ambulance crew']]]

gives the semantic of the sequential flow of three atomic actions *'The resource allocator identifies the resource to be mobilised. Then, the resource allocator enters the mobilisation decision in the CAD system, and the CAD system sends the mobilisation order to the relevant ambulance crew'*.

The *constraint* is a sentence semantic pattern that relates a condition to any other constrained sentence or clause. The semantics of the constrained sentence can be itself analysed through semantic patterns. Similarly, the *repetition* relates a condition to a repeated atomic action or flow of actions expressed by a clause or by a sentence. Thus, the semantics of the repeated element can be itself recursively analysed.

The same relationship between sentence semantic patterns and flows of actions applies to the sequence, concurrency, constraint and repetition patterns. The concurrency sentence semantic pattern provides the semantic relationship of co-occurring actions, like *‘While it sends the mobilisation decision to the relevant ambulance crew the CAD system updates the availability database’*. The repetition pattern identifies actions that are repeated under a certain condition, such as in *‘the resource allocator enters a mobilisation decision until it is valid’*.

3.2 Semantic patterns and surface structures

Contrarily to semantic patterns, surface structures allow several representations of the surface level representation of a piece of text. Indeed, at the surface level, there are often numerous ways to express the same knowledge. The syntactic rules of natural language manage the sequencing of words, and several sequencing are allowed for a sentence without modifying its deep meaning. We call *structure* the different surface representations of a semantic pattern.

Consider for example the communication clause semantic pattern :

Communication (V) [Agent ; Object ; Source ; Destination],

where the agent is the active entity that initiates or controls the action, the object is the passive entity which is communicated, and the source and destination are respectively the origin and destination of the communicated object. The structures, described below, identify several ways to express communications. Either through verb groups which main verb is a verb of communication (such as to read, to get, to give, to move), or through noun groups which main noun corresponds to the substantive form of verbs of action (the giving, the moving, etc). These structures are the ones that can be expected from the prose of the scenario author.

S1 : [[NG](Subject)$_{\text{Ag}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$ [*from* NG](Complement)$_{\text{So}}$ [*to* NG](Complement)$_{\text{Dest}}$](VG active)$_{\text{Communication}}$
S2 : [[NG](Subject)$_{\text{Ag}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$ [*from* NG](Complement)$_{\text{So}}$](VG active)$_{\text{Communication}}$
S3 : [[NG](Subject)$_{\text{Ag}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$ [*to* NG](Complement)$_{\text{Dest}}$](VG active)$_{\text{Communication}}$
S4 : [[NG](Subject)$_{\text{Ag+So}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$ [*to* NG](Complement)$_{\text{Dest}}$](VG active)$_{\text{Communication}}$
S5 : [[NG](Subject)$_{\text{Ag+So}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$](VG active)$_{\text{Communication}}$
S6 : [[NG](Subject)$_{\text{Ag+Dest}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$ [*from* NG](Complement)$_{\text{So}}$](VG active)$_{\text{Communication}}$
S7 : [[NG](Subject)$_{\text{Ag+Dest}}$ [Verb]$_{\text{Communication}}$ [NG](Complement)$_{\text{Obj}}$](VG active)$_{\text{Communication}}$
S8 : [[NG](Subject)$_{\text{Ag+Obj}}$ [Verb]$_{\text{Communication}}$ [*from* NG](Complement)$_{\text{So}}$ [*to* NG](Complement)$_{\text{Dest}}$](VG active)$_{\text{Communication}}$
...
S11 : [[NG](Subject)$_{\text{Obj}}$ [*be* Verb]$_{\text{Communication}}$ [*by* NG](Complement)$_{\text{Ag}}$ [from NG](Complement)$_{\text{So}}$ [*to* NG](Complement)$_{\text{Dest}}$](VG passive)$_{\text{Communication}}$
S12 : [[NG](Subject)$_{\text{Obj}}$ [*be* Verb]$_{\text{Communication}}$ [*by* NG](Complement)$_{\text{Ag+So}}$ [*to* NG](Complement)$_{\text{Dest}}$](VG passive)$_{\text{Communication}}$
S13 : [[NG](Subject)$_{\text{Obj}}$ [*be* Verb]$_{\text{Communication}}$ [*to* NG](Complement)$_{\text{Dest}}$ [*by* NG](Complement)$_{\text{Ag+So}}$](VG passive)$_{\text{Communication}}$
S21 : [[[N substantive form of action Verb](Main Noun)$_{\text{Communication}}$ [*of* NG](Epithet)$_{\text{Obj}}$ [*to* NG](Epithet)$_{\text{Dest}}$ [*by* NG](Epithet)$_{\text{Ag+So}}$](NS)$_{\text{Communication}}$

For example, the atomic action *‘the resource allocator enters a mobilisation decision in the CAD system’* (here expressed using structure S4) can also have the surface representations :

(S4) *‘The resource allocator is entering a mobilisation decision in the CAD system’*,

(S12) *'A mobilisation decision is entered by the resource allocator in the CAD system'*,
(S13) *'A mobilisation decision is entered in the CAD system by the resource allocator'*,
(S21) *'The entering of the mobilisation decision in the CAD system by the resource allocator'*.

Such structure naturally occurs in sentences such as : *'the the entering of the mobilisation decision in the CAD system by the resource allocator occurs before the sending of the mobilisation order'*.

To each clause semantic pattern and to each sentence semantic pattern is associated a list of surface structures which provide numerous ways of expressing atomic actions, and flows of actions in a scenario. Some structures allow to express incomplete knowledge in reference to the semantic patterns. For example the structure S5 corresponds to the sentence *'the CAD system sends the mobilisation decision'* in which the destination is missing. In such a case, the instantiation of the communication pattern identifies the missing element by marking the non-instantiated case with a '?' flag, as in :

Communication (send) [Agent : *'the CAD system'* ; Object *: 'the mobilisation decision'* ; Source : *'the CAD system'* ; Destination : ?]

3.3 Structures, patterns, and scenario model

Let us sum up the relationships between the scenario model, structures and semantic patterns. Each structure has a semantics defined in a semantic pattern. The semantics of sentence structures is in sentence semantic patterns, and the semantics of clause is in clause semantics patterns. The other way round, one semantic pattern can be expressed following several different surface level structures.

In addition, each concept of the scenario model has a semantic identified in a semantic pattern. For example atomic actions correspond to the communication and action clause semantic patterns. The patterns of sequence, constraint, concurrency and repetition identify the flows of actions of the scenario model ; that is to say respectively the sequence, constraint, concurrency, and iteration. The semantic patterns being expressed using pre-defined surface structures, each concept of the scenario model can be expressed in a restricted natural language. As an example, the following piece of scenario specification :

Atomic action :
Name : 'enter'
From Agent : 'the control assistant'
To Agent : 'the CAD system'
Parameter : 'the details of the call'

corresponds to the pattern instance :

Communication ('enter') [Agent : 'the control assistant' ; Object : 'the details of the call' ; Source : 'the control assistant' ; Destination : 'the CAD system']

This communication action clause semantic pattern can be expressed using the structure :

[['The details of the call'](Subject)$_{Obj}$ *['are entered'](Main Verb)*$_{Communication}$ *['by the control assistant'](Complement)*$_{Ag+So}$ *['in the CAD system'](Complement)*$_{Dest}$*](VG passive)*$_{Communication}$

which corresponds to the clause :

'The details of the call are entered by the control assistant in the CAD system'.

Identifying the concepts of the scenario model from natural language is a two stage process which requires first the semantic analysis of the text and then the mapping of the resulting semantic patterns onto the concepts of the scenario model. However, the conceptualisation of scenarios is possible only if the scenarios are well written. These two aspects are dealt with in the following sections.

4. The authoring process

In the CREWS goal-scenario approach, the activity of scenario authoring is complementary to the goal discovery (see Figure 5). Both can be performed iteratively, until full completeness, validity and agreement. The goal discovery is described in [21]. The scenario authoring is in two stages : the writing of the scenarios, and the correcting of scenarios. Once written and corrected, scenarios can be used for discovering new goals or for negotiating the requirements.

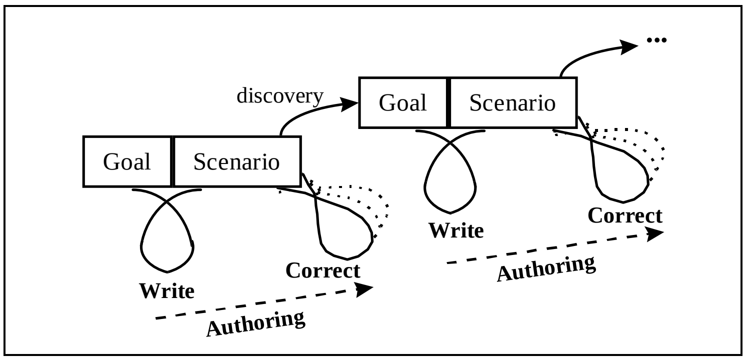

Figure 5 : Overview of the scenario authoring process.

To guide the Scenario Author (SA) in the writing of scenarios, we propose *writing guidelines* (Figure 6). When he / she wants to write a scenario, the SA has already defined a goal that the scenario will illustrate. Thus, the guidelines drive the SA in illustrating a given goal by a scenario. The guidelines consist in a list of advice on how to write a scenario and what to put in the scenario. We expect that the quality of the scenarios produced improves when the guidelines are correctly applied. However, the guidelines are mandatory : they can be followed or not. Thus, rules are proposed to verify and improve the quality of scenarios with respect to the guidelines.

To guide the SA to correct scenarios, we propose a set of *correcting rules* (Figure 6). The correcting of scenarios involves conceptualisation, clarification and completion. The *conceptualisation* of scenarios is performed according to the relationships between the scenario structure, its semantics, and the model of scenarios, respectively presented in the sections 3 and 2. Conceptualisation is necessary to enact other verification rules. It involves structure analysis, semantics analysis, and model instantiation. The *clarification* rules help the SA identifying and correcting ambiguous actions. The *completion* rules guide the SA in the search of missing elements in the scenario, for example atomic actions with unprecise parameters, etc.

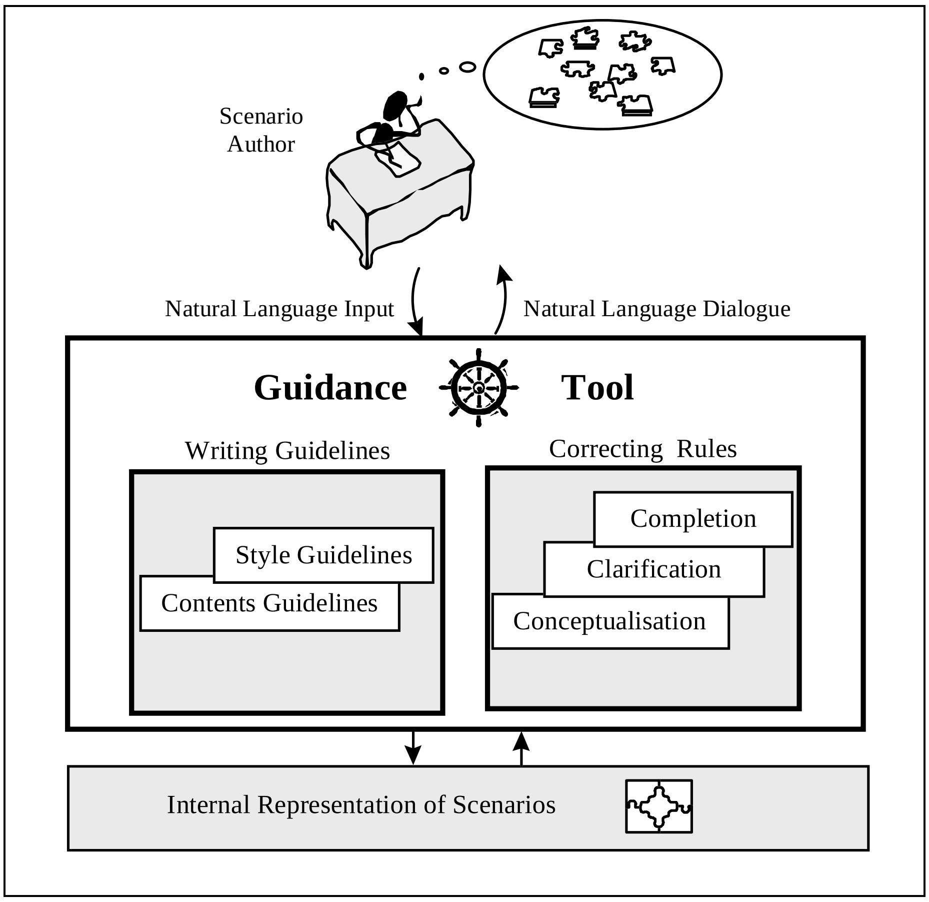

Figure 6 : Overall architecture of the scenario authoring process.

Our guidelines for writing scenarios are presented in the next section. We shall describe and illustrate the correcting rules in section 6.

5. Guidelines for scenario writing

The writing guidelines are of two types : style and contents guidelines. *Style guidelines* provide recommendations on the style of the expected prose. *Contents guidelines* advise the SA on the expected contents of his / her prose. Style and contents guidelines are complementary. A scenario can be written in a good style but have a wrong contents. For example it can be out of the subject of the problem domain. The other way round, the contents of a scenario can be valid, but the scenario written in a bad style, and lead to erroneous interpretations. On the one hand, contents guidelines are adapted to the scenario model of the section 2 and to the expected text semantics presented in section 3. On the other hand, the style guidelines aim at helping the user adopting good style, in conformance with the expected text structure presented in section 3.

The style and contents guidelines have the form of plain text. They can be prompted to the scenario author on demand while writing down a scenario.

Contents guidelines

- The expected scenario prose is a description of a *single course of actions*. This course of actions should be an illustration of fulfilment of your goal. Alternative scenarios and exceptions, should be described separately.

- You should *describe the course of actions you expect*, not the actions which are not expected, impossible, not relevant with regard to the problem domain. Therefore, *avoid modal verbs* such as should, could, can, may, etc. In addition, *avoid negation* : not, never.
- A course of actions describes *sequentially ordered actions* (in other word, a kind of story). Any action should be directed from an agent to another agent, and apply on a parameter.
- *State explicitly the assumptions* allowing to differentiate your scenario from another. In particular, you should not forget to state the cause of an exception. For example use the template proposed in the style guidelines for flow conditions.
- If there are loops, they should be restricted by conditions, in order that *the number of occurrences of the repetition is finite*. For example use the template proposed in the style guidelines for repetitions.
- You should stay at *one level of abstraction all along the scenario*.

Style guidelines

- Any atomic action should be stated in a separate clause as *a verb together with one or several parameters*. Therefore, an atomic action can be expressed as one of the following clause templates :

  <Agent> <Action> <Parameter>
  ex. : '*<The system> <checks> <the incident location>*'
  <Agent1> <Action> <Parameter> from <Agent2>
  ex. : *'The controller reads the incident location from the display'*
  <Agent1> <Action> <Parameter> to <Agent2>
  ex : *'The system displays an error message to the controller'*
- Be careful on the fact that *the scenario model does not include circumstances*. Thus, atomic actions should not include time, duration, location, manner, etc.
- In order to avoid forgetting one of the agents, *make use of the active voice*.
- *Avoid* making explicit references to the scenario model.
- The actions should be preceded by their *reference number*. A reference number should be unique in a scenario, even if the corresponding action occurs twice in the scenario.
- *Respect the ordering* of the story you are writing down. Thus, avoid flash backs and forward references. Express explicitly sequences. At least, use carriage return for each new action occurring after an other. Moreover, if you want to express a sequence of an action repeated on two different objects, you should express the two actions explicitly, and avoid factoring the objects (e.g. 'the system checks the name and the password'). Similarly, if an object participates to two consecutive actions, avoid its factorisation (e.g. 'the password is checked and modified').
- Flow conditions should be explicitly *stated at the moment where they influence the course of actions*. Therefore, a flow condition can be expressed as in one of the templates :

  If <condition> Then ...
  Repeat... Until <condition>

  Make use of indentation in order to make explicit which part of the scenario is constrained by the flow conditions.
- You should *avoid the use of anaphoric references* such as « he », « she », « it » « his »or « him ». Instead, you should use nouns, as defined in the following guideline.
- You should *be consistent in the scenario terminology*. Therefore, avoid the use of synonyms (one object with two different names) and homonyms (two different objects with the same name). The same object or action should be named identically throughout all the scenario text. In particular it can occur that you want to describe two different agents of the

same family (for example two different ‘customers’). Then, be careful to distinguish the two agents (for example, use the names ‘customer C1’ and ‘customer C2’).

At the moment of writing a scenario, the scenario author can use the help provided by the guidelines. The style and contents guidelines are independent. Thus, the scenario author can use both style and contents guidelines, or only one of the two kinds of guidelines. The scenarios produced following the guidelines should be of acceptable quality ; in other word, their syntax and their semantics should correspond to the ones expected through the scenario model, the structures, and the semantic patterns. We are currently evaluating empirically the help provided by the guidelines.

In addition, we propose a number of rules to identify violations of the guidelines, and guide the scenario author in correcting his / her scenarios. These rules are presented in the following section.

6. Rules for correcting scenarios

The scenario author writes a scenario following more or less the guidelines he is provided with. Thus, it is necessary to verify that the scenarios are well-written. In this section, we focus on the correcting of scenarios.

The correcting of scenarios is performed by a set of rules as presented below. The quality of scenarios is evaluated in reference to : the structure, the semantics and the contents presented in the previous sections, and provided to the SA through the guidelines. Thus, the rules aim at identifying conflicts between a scenario, and a guideline. A rule is described intentionally with a goal, and a body. The rule goals recall the main objectives of the rules. The bodies consist in steps. These steps aim at

- identifying suspect situations,
- proposing solutions to modify the scenarios,
- acquiring complementary information about the scenario contents, and
- improving the scenarios.

The rules shall be explained and illustrated with the example of the London Ambulance Service case study.

Clarification rule CLA1

```
Goal : clarify the scenario structure.

Body : for each sentence of the scenario do
    (1) Analyse the syntax of the sentence.
    (2) If the sentence syntax does not fit a structure then
        (3) If the error is due to an unexpected adverb then
            (4) Propose the SA to rephrase the sentence or to remove this adverb, and to
            build a requirement to quantify or qualify the corresponding action.
        (5) If the error is due to an unexpected complement then
            (6) Propose the SA to rephrase the sentence or to remove this complement,
            and to build a requirement to situate the corresponding action.
        (7) If the error is due to an unexpected modal verb then
            (8) Propose the SA to rephrase the sentence or to remove this modal verb,
            and to build a requirement to value the necessity of the corresponding
            action.
        (9) If the error is due to another unexpected sentence construction then
```

```
(10) Propose the SA to rephrase the sentence or to remove it.
```

As underlined in the guidelines, the scenario model defines borders to the scenario expression. Indeed, elements of the language such as circumstance complements, modal verbs, or negation, do not have an equivalent in the scenario model. The structures proposed in section 3 define which sentence structure is acceptable, and which should be clarified. Let's assume for example that the SA has used the guidelines to write the scenario *'Allocate incidents to an ambulance crew'* presented in Figure 2, and consider the actions :

7. '*The resource allocator identifies the resource to be mobilised',*
8. '*He must enter his mobilisation decision within two minutes',* and
9. '*The system sends rapidly the mobilisation order to the relevant ambulance crew*'.

Analysing these clauses, the rule identifies that the clause (7) fits to one of the structures proposed to express an atomic action in a scenario. On the contrary, the clause (8) includes both a modal verb *'must',* and a circumstantial complement of time *'within two minutes'.* However, as underlined in the guidelines none of them appear in the structures corresponding to the scenario model. Therefore, the SA is proposed to remove the sentence, or to rephrase it and create a new requirement. As it is necessary to keep the atomic action, the SA decides to rephrase it and it becomes :

8. *'He enters his mobilisation decision'.*

Similarly, the action number (9) includes the adverb 'rapidly'. The SA rephrases it :

9. '*The system sends the mobilisation order to the relevant ambulance crew*'

The two atomic actions are now expressed correctly in reference to the scenario model. In addition, the SA author identifies new requirements on the use of the CAD system :

R1 : 'The resource allocator must enter a mobilisation decision within the two minutes that follow the moment when he is provided with the details of an incident'.

R2 : 'The time between the moment when a mobilisation order is sent by the CAD system and the moment when it is received by the ambulance crew must be inferior to thirty seconds'.

Clarification rule CLA2

```
Goal : clarify ambiguous terms of the scenario
Body : for each sentence of the scenario do
    (1) Analyse the structure of the sentence
    (2) For each element identified in the structure do
    (3) If the element contains an anaphoric reference then
        (4) Suggest the SA to replace the reference by a term consistent with the scenario
            terminology.
```

Anaphoric reference are references to other parts of the text. Typically, pronouns such as 'he', 'she', 'it', 'him', 'them', and possessive adjectives as 'my', 'your', 'his' etc, are anaphoric references. Not only are they bringing inconsistency in the use of the terminology (the same object is referred to by a name and by a pronoun), but also they are ambiguous and can be misinterpreted.

Let's take the example of Figure 2 (b). This text includes 3 ambiguous references :

'It' in 4. *'It decides which division is going to deal with the incident',* and

‘He’ and ‘his’ in 8. *‘He must enter his mobilisation decision’*.

The SA is asked to replace these anaphoric references by less ambiguous terms that would be consistent with the terminology used in the remaining of the scenario. The SA decides to perform the following corrections : replace ‘it’ by ‘the CAD system’ in action 4, replace ‘he’ by ‘the resource allocator’ in action 8, and remove ‘his’ from action 8. The two actions become :

4. *‘The CAD system decides which division is going to deal with the incident’*, and
8. *‘The resource allocator enters the mobilisation decision’*.

Completion rule COM1

```
Goal : complete atomic actions

Body : for each sentence of the scenario do
    (1) Identify the structure of the sentence.
    (2) If the structure of the sentence is complex then for each sub-clause do
        (3) Identify the appropriate action semantic pattern.
        (4) Map the elements of the clause on the cases of the semantic pattern.
        (5) If the semantic pattern is incompletely mapped then
            (6) Ask the SA to complete the action with the missing element.
```

This rule aims at correcting the sentences of the scenario which instantiate incompletely a semantic pattern. Indeed, a semantic pattern which is not fully instantiated means an incomplete clause or sentence, that is to say in the terms of the scenario model an incomplete action. Therefore, the rule instantiates semantic patterns from the given scenario, and, for any incompletely instantiated pattern, the SA is asked to provide the missing element.

As mentioned in section 3.2, when a pattern can not be fully instantiated from a sentence or a clause structure, the missing element is replaced by the ‘?’ mark. Considering the scenario of Figure 2, after the modifications brought by the rules CLA1 and CLA2, the rule COM1 finds the following incompletely instantiated semantic patterns :

1 : Communication (*enter*) [Agent : ? ; Object : *‘the details of the call’* ; Source : ? ; Destination : *‘the CAD system’*]
5 : Communication (*display*) [Agent : ? ; Object : *‘the resource and incident details’* ; Source : ? ; Destination : *‘the division resource allocator’*]
8 : Communication (*enter*) [Agent : *‘the resource allocator’* ; Object *: ‘the mobilisation decision’* ; Source : *‘the resource allocator’* ; Destination : ?]

The scenario author is demanded to ‘fill the blanks’. For example, the rules asks the SA to ‘Supply an agent an a source for the action 1 : *The details of the calls are entered in the CAD system’*. The SA author can rephrase partially or fully the sentence. The result of completing the incomplete actions for the whole scenario text leads to the new scenario presented below. The completed actions are shown in bold.

1. **The control assistant enters the details of the call in the CAD system.**
2. If the call is not in duplicate, then
    3. The CAD system locates the incident place.
    4. **The CAD system decides which division is going to deal with the incident.**
    5. The resource status and incident details are displayed by the CAD system to the division resource allocator.
    6. If an ambulance and a crew are available at the division, then
        7. The resource allocator identifies the resource to be mobilised.

8. **The resource allocator enters the mobilisation decision in the system.**
9. The system sends the mobilisation order to the relevant ambulance crew.

Clarification rule CLA3

```
Goal : clarify inconsistent terms of the scenario
Body : for each sentence of the scenario do
    (1) Identify the structure of the sentence.
    (2) For each term T identified in the structure, do
        (3) Check the term T against the scenario glossary
        (4) If the term T is not in the glossary, then
            (5) If the term T has a synonymous T' in the glossary, then
            (6) Suggest the SA to replace T' by T in the glossary and in the remaining of
                the scenario, or to replace T by T' in the scenario.
            (7) Else, propose the SA to add it as a new term of the glossary.
```

This rule aims at verifying that the objects of the scenario are referred to with a unique name. First, the structure of the text is analysed. Then each element identified in the text structure is considered in turn to build a scenario glossary. First, the element is compared to each entry in the glossary. If a synonymous is found, the SA is suggested either to replace the term in the glossary and in the remaining of the scenario by the new term, or to replace the new term to the term already accepted in the glossary. If no synonymous is found, the SA has the possibility to complete the glossary with a new term.

For example, let's assume that the SA has already partly checked the consistency of terminology in the scenario presented above. The terms identified from the scenario actions 1 to 6 are set in the following glossary together with the reference of the actions in which they appear :

- control assistant (1)
- CAD system (1, 3, 4, 5)
- division (4, 6)
- incident (4)
- incident details (5)
- ambulance (6)
- details of the call (1)
- incident place (3)
- call (2)
- resource status (5)
- division resource allocator(5)
- crew (6)

Following on, the ruleCLA3 has now to consider the actions (7), (8), and (9).

The action (7) references two objects : the 'resource to be mobilised', and the 'resource allocator'. The former does not appear, and has no equivalent in the scenario. Therefore this term can be added into the scenario glossary. On the contrary, the latter is mentioned as the 'division resource allocator' in the glossary. Thus, the rule proposes to the SA to modify this term, or to update the glossary and the beginning of the scenario. The SA author decides to modify the action (7) which becomes :

7. *'The division resource allocator identifies the resource to be mobilised'*.

The action (8) of the scenario presenting the same error, the term is also modified in this action. In addition, the actions (8) and (9) refer to the *'system'*, which already exists as the 'CAD system' in the glossary. The SA decides to modify both actions, in order to call coherently the 'CAD system' all along the scenario.

Going on with the scenario, the rule shows the SA, that the 'resource to be mobilised', the 'mobilisation decision', and the 'mobilisation order' refer to the same thing. The SA decides to name it the ‘scenario order’ all along the scenario. Once modified, the actions (7), (8), and (9) become :

7. *‘The division resource allocator decides of a mobilisation order’*,
8. *‘The division resource allocator enters the mobilisation order in the CAD system’*,
9. *‘The CAD system sends the mobilisation order to the relevant ambulance crew’*.

Conceptualisation rule CON1

```
Goal : conceptualise a textual scenario
Body : for each sentence of the scenario do
    (1) Analyse the structure of the sentence
    (2) Identify the semantic pattern corresponding to the structure.
    (3) If the pattern is a sentence pattern, then
        (4) Apply again the steps 1 and 2 on its parameters.
    (5) Instantiate the parameters of the semantic pattern.
    (6) Map the semantic pattern on the concepts of the scenario model
    (7) If the pattern can not be mapped, then
        (8) Propose the SA to remove the corresponding text from the scenario.
```

Once the semantics of the scenario is analysed, a mapping is performed from the scenario semantics to the scenario model. The mapping step identifies semantic patterns to actions of the scenario model. For example, the sequence semantic pattern is mapped onto sequence flow of action. The communication action clause pattern is mapped, if one of the directions is the same as the agent, onto an atomic action. A formal definition of the mapping is provided in [20].

As shown in the step (7) of the rule body, some patterns can not be mapped onto the scenario model. For example, the scenario model shows that flow of actions are composed of actions. Therefore, a sequence including states can not be mapped onto the scenario model. When it occurs that the SA includes states in a flow of actions, the scenario does not respect the scenario model, and an error is detected. Similarly, by definition, a scenario is a single path. Therefore, a sequence including a constraint followed by a action is erroneous in regard to the scenario model. Indeed, this combination identifies two paths : the action occurs, were the condition associated to the constraint verified or not.

In order to illustrate the conceptualisation of a textual scenario, let’s take the example of the action (8).

8. ‘The division resource allocator enters the mobilisation order in the CAD system’.

This clause instantiates the communication clause pattern as follows :

Communication (enter) [Agent : *‘division resource allocator’* ; Object *: ‘mobilisation order’* ; Source : *‘division resource allocator’* ; Destination : *‘CAD system’*]

One can see that the agent and the source of the communication are the same. The clause can thus be conceptualised as an atomic action. The name of the action is given by the verb of the clause, the ‘from’ agent by the source, the ‘to’ agent by the destination, and the parameter by the object. The internal representation of the action is thus :

Atomic action :
  Name : *‘enter’*
  From Agent : *‘division resource allocator’*

To Agent : *'CAD system'*
Parameter : *'mobilisation order'*

Once fully conceptualised, this formal representation of the scenario can be used to generate graphical representations of the scenario (as a sequence diagram of the UML notation [1]), to make fully automated reasoning on the scenario, or to discover new goals, as shown in [21].

7. Conclusion

Very few approaches say how to author textual scenarios. Based on a formal conceptual model of scenarios, this paper aims at guiding the process of scenario authoring. This guidance emphasises two stages : writing and correcting.

To help the scenario author writing textual scenarios, we propose a set of guidelines. We are currently evaluating these guidelines by empirical evaluations with students and engineers, in collaboration with the City University (UK). In addition, examples of application of the guidelines have been already explored with the ATM case study [20], and with a real case study borrowed to an Electricity Company [13].

Besides the support of scenario writing, we propose a set of rules to guide the clarification, completion and conceptualisation of scenarios. As for the guidelines, the linguistic foundation for these rules is the Case Grammar which is tailored for the scenario conceptual model. The grammar defines linguistic structures and semantic patterns lying on the concepts of the scenario model. The former identify the numerous ways to express atomic actions, flows of actions, conditions in natural language. The latter expresses the semantics of the scenario contents. The grammar permits us to go beyond the simple solution of sentence templates by providing tools to support the writing and correcting of scenarios.

The process is illustrated with the case study of the Computer Aided Despatch system of the London Ambulance Service. However we focused on the linguistic support and restricted to the CAD-user interactions. Our current work consists in extending the guidance rules to support the discovery of goals, themselves used as input for the authoring process. In addition, we are extending the Visual Basic - PROLOG implementation to support the entire set of rules.

8. References

[1] C. Ben Achour, *Linguistic Instruments for the Integration of Scenarios in Requirements Engineering*. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona, pp. 145-156, June 1997.

[2] B. Boguraev and K. Spark-Jones, *A Note on a Study of Cases*. In *Computational Linguistics*, Vol 13, n° 1-2, pp. 65-68, 1987.

[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, http://members.aol.com/acockburn/papers/usecases.htm, 1995.

[4] CREWS Team, *The CREWS glossary*. CREWS report, http ://SUNSITE.informatik.rwth-aachen.de/CREWS/reports.htm, 1998.

[5] S.C. Dik, *'The Theory of Functional Grammar, part I : the Structure of the Clause'*. Functional Grammar Series, Fories Publications, 1989.

[6] T. Erickson, *Notes on Design Practice: Stories and Prototypes as Catalysts for Communication*. in ‘Scenario-Based Design: Envisioning Work and Technology in System Development’, Ed J.M. Carroll, 1995.

[7] C. Fillmore, *The case for case*. In E.Bach, R. Harms (eds.) ‘Universals in linguistic theory’, Holt, Rinehart and Winston Publishing Company, pp. 1-90, 1968.

[8] A. Finkelstein, J. Dowell, *A Comedy of Errors : the London Ambulance Service Case Study*. Proceedings of the 8th International Workshop on Software Specification and Design, Germany, 1996.

[9] I. Jacobson, M. Ericsson and A. Jacobson, *‘The Object Advantage, Business Process Reengineering with Object Technology’*. Addison-Wesley Publishing Company, 1995.

[10] M. Jarke, K. Pohl, P. Haumer, K. Weidenhaupt, E. Dubois, P. Heymans, C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyte, A. Sutcliffe, N.A.M. Maiden and S. Monicha, *Scenario Use in European Software Organisations - Results from Site Visits and Questionnaires*. CREWS deliverable W1: Industrial problem capture Working Group, 1997.

[11] M. Kyng, *Creating Contexts for Design*. In John M. Carroll (ed.), ‘Scenario-Based Design: Envisioning Work and Technology in System Development’, John Wiley and Sons, pp. 85-107, 1995.

[12] B. Lawrence, *Do you really need Formal Requirements ?* IEEE Software 13(2) :20, 1996.

[13] 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.

[14] K. Pohl, P. Haumer, *Modelling Contextual Information about Scenarios*. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ’97, Barcelona, pp.187-204, June 1997.

[15] C. Potts, K. Takahashi and A.I. Anton, *Inquiry-based Requirements Analysis*. in IEEE Software 11(2) (1994) 21-32.

[16] 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.

[17] N. Prat, *Goal Formalisation and Classification for Requirements Engineering*. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ’97, Barcelona, pp. 145-156, June 1997.

[18] B. Regnell, K. Kimbler, A. Wesslen, *Improving the Use Case Driven Approach to Requirements Engineering*. Second IEEE International Symposium On Requirements Engineering, (York, England), I.C.S. Press, pp. 40-47, March 1995.

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

[20]. C. Rolland, C. Ben Achour, *Guiding the Construction of Textual Use Case Specifications*. Data & Knowledge Engineering Journal, Vol 25, N°1-2, pp 125-160, March 1998.

[21] C. Rolland, C. Souveyet, C. Ben Achour. *Guiding Goal Modelling Using Scenarios.* Submitted to the TSE Journal, special issue on scenarios, 1998.

[22] K.S. Rubin and A. Goldberg, *Object Behaviour Analysis*. Communications of the ACM, 35(9), pp. 48-62, 1992.

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

[24] K. Ryan, *The Role of Natural Language in Requirements Engineering*. Proceedings of the IEEE Int. Symposium on RE, San Diego California, pp. 240-242, 1993.

[25] R.C. Schanck, *Identification of conceptualisations underlying natural language*. In 'Computer models of thought and language', R.C. Shanck, K.M. Colby (eds), Freeman, San Francisco, pp. 187-247, 1973.

[26] R. Simmons, *Semantic Networks : their Computation and Use for Understanding English Sentences*. In 'Computer Models of Thought and Language', R.C. Schanck, K.M. Colby (eds), Freeman, San Francisco, pp. 63-113, 1973.

[27] K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer, CREWS Team, *Scenario Usage in System Development : a Report on Current Practice*. Proceedings of the Third International Conference on Requirements Engineering, ICRE'98, Colorado Springs, USA, 1998.

[28] Y. Wilks, *Good and Bad Arguments about Semantic Primitives.* Report n° 42, Department of Artificial Intelligence, University of Edinburgh, 1977.