Requirements Eng (2005) 10: 1–21
DOI 10.1007/s00766-003-0186-9

ORIGINAL ARTICLE

Julio Cesar Sampaio do Prado Leite
Jorge Horacio Doorn · Graciela D. S. Hadad
Gladys N. Kaplan

Scenario inspections

Received: 15 February 2002 / Accepted: 19 September 2003 / Published online: 29 January 2004

Abstract Scenarios help practitioners to better understand the requirements of a software system as well as its interface with the environment. However, despite their widespread use both by object-oriented development teams and human–computer interface designers, scenarios are being built in a very ad-hoc way. Departing from the requirements engineering viewpoint, this article shows how inspections help software developers to better manage the production of scenarios. We used Fagan's inspections as the main paradigm in the design of our proposed process. The process was applied to case studies and data were collected regarding the types of problems as well as the effort to find them.

Keywords Inspections · Requirements verification · Scenarios · Scenario quality

1 Introduction

The word *scenario*, defined as "the plot of a motion picture" in the Merriam-Webster Dictionary, has been used in different disciplines. However, the software community has provided a new meaning for scenario: "a description technique that is both process-focussed and user-centric". The literature has been prolific in providing different representations, interpretations, and processes to implement the scenarios concept [1, 2], but the major emphasis is on narrative description and on the usage of examples or cases. In the object-oriented community [3, 4], the accepted term for these descriptions is *use cases*, the word scenario being used to detail conditions of a particular use case.

CREWS (Cooperative Requirements Engineering with Scenarios), a European Esprit project on scenarios [5, 6], conducted a survey of research and industrial practice. The survey pointed out that although several representations and techniques have been proposed and used in industrial settings, there is a lack of knowledge in managing scenario construction and evolution. This lack of precision about when and how scenarios should be used has spread to the engineers who are using these techniques in the field. Thus, most developers see scenario creation as a craft more than an engineering task. The industrial part of the survey [6] clearly showed this lack of discipline in producing scenarios. It also pointed out the necessity of more detailed definitions about scenario construction as an unavoidable factor to increase their use and productivity in real situations.

We believe that a scenario description is a means to elicit application knowledge as well as to register elicited information. As such, it acts as a complementary description in the process of requirements definition. In this context, our work [7] has paid special attention to the lack of processes for scenario construction. One of the aspects we researched was a quality assurance strategy for scenarios. How can one guarantee that scenario descriptions are of good quality?

We meet this challenge by relying on a review process. We understand that one of the most effective methods to achieve productivity with quality in producing software is the usage of a review process. The idea is simple and several reports confirm its usefulness.

J. C. S. P. Leite (✉)
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio,
Rua Marquês de São Vicente 255,
22451-041 Gávea-RJ Rio de Janeiro, Brazil
E-mail: julio@inf.puc-rio.br
Fax: + 55-21-31141530

J. H. Doorn
INTIA, Universidad Nacional del Centro de la Provincia
de Buenos Aires, FRBA,
Universidad Tecnológica Nacional,
Pinto 399, 7000 Tandil, Argentina

G. D. S. Hadad
FRBA, Universidad Tecnológica Nacional,
LINTI, Universidad Nacional de La Plata,
Buenos Aires, Argentina

G. N. Kaplan
FRBA, Universidad Tecnológica Nacional,
LINTI, Universidad Nacional de La Plata,
Buenos Aires Argentina

Reviewing is an old tradition in the publication industry and has been in practice for centuries. Some people write and others read the written document for the purpose of correcting mistakes and shaping the text according to predefined policies.

In software engineering, several authors have proposed reviews: the author/reader cycle of SADT [8], the structured walkthrough [9, 10], design reviews [11], and inspections [12, 13]. Only more recently [14, 15, 16, 17] have software developers become convinced that reviews are a necessity. Nonetheless, most of the processes in use today still base their quality assurance on testing.

In our scenario construction process [7], we chose inspections as a way of assuring quality for scenarios. This idea of applying inspections to improve scenario quality is intuitive and has been applied before [18]. In the industrial CREWS survey [6], it was reported that nine out of 15 projects had used some kind of review process. Our proposal addresses the major weaknesses of the inspection process [11, 19] by focussing attention on the preparation step. Our preparation step uses a reading technique known as "scenario-based reading"[1] [19], which provides a strong systematization for this step.

Our contribution is to provide a detailed verification process for "scenario analysis" based on a reading technique. We believe that our strategy will help researchers and practitioners addressing aspects of quality assurance for very early software representations. It is important to stress that we are reporting on a proposal designed for a specific scenario construction process (Sect. 2.1), but which is general enough to be adapted to other scenario-based methods [20]. Our proposal for quality assurance, called "scenario analysis", uses a combination of verification and validation steps. However, our paper is limited to the verification step. It is independent of other quality assurance methods, but may, in addition, be used in combination with other analysis processes. Verification is performed among developers, contrary to validation, which requires the involvement of customers. By verifying scenarios we are improving situation descriptions, and as such improving the overall requirements process.

This paper aims to contribute to issues that may, in the future, be handled semi-automatically with proper tools. Our focus is on method and its use. We foresee that tools will be necessary to make our processes more efficient and easy to use.

[1] As it has been well noted by Regnell in [17, p. 142]: "There is considerable risk for terminology confusion here, as the term *scenario* also is used within requirements engineering to denote a sequence of events involved in an envisaged usage situation of the system under development. A *use case* is often said to cover a set of related (system usage) scenarios. In scenario-based reading, however, the term scenario is a meta-level concept, denoting a procedure that a reader of a document should follow during inspection." This note clearly shows the confusion; when we use the term scenario-based we are saying what Porter defined in [19]: "Scenarios – collections of procedures for detecting particular classes of faults."

Fig. 1 The scenario representation (from [7])

It is important to say that our proposal is performed before the validation of scenarios and should be done by requirements engineers. We focus our attention on a detailed description of a verification process. Although we include data from case studies, this paper does not describe an experiment. We list the case studies to provide for the reader an idea of how the method is used, the effort required to use it, as well as an indication of what could be improved. The method, as described here, includes the feedback we received from the case studies detailed in Sect. 5.

The article is structured as follows: our scenario representation schema along with a summary of our scenario construction process (Sect. 2), a review of inspections (Sect. 3), details of the proposed strategy (Sect. 4), the case studies (Sect. 5), and conclusions (Sect. 6).

2 Scenarios

We adopt a requirements engineering view of scenarios. We are using the term *scenario* to name descriptions of situations [21], and this is in accordance with its use by the human–computer interface community [22]. To us, scenarios describe situations taking into account usage aspects, allowing us: to define the problem, to unify criteria, to gain clients/users compromise, to organize the involved details, to train new participants [22], and to provide an anchor for traceability [1]. The use of scenarios as a technique to understand the problem to be solved using a software system has been recommended by several authors [21, 23, 24].

The kind of knowledge we are dealing with is what Rolland et al. classified as *organizational context* [5], which aims at the "broad picture of how the work gets done". Our conjecture, confirmed by several case studies as well as by the literature, is that scenarios provide an attractive communication means among stakeholders in the universe of discourse (UofD). Here scenarios become important, since they may hold much information in a way that stakeholders could recognize.

A scenario is a partial description of the application behaviour that occurs at a given time within a certain geographical context, i.e. a situation [21]. We use a natural language structured description as its basic representation. To this definition, we added two concepts [1]:

- A scenario evolves as we progress in the software construction process.
- Scenarios are naturally linked to the Language Extended Lexicon (LEL) (see Sect. 2.2).

To us, a scenario describes a particular situation of the UofD showing the main course of action but also including variations or possible alternative cases. The use of natural language, in particular the client/user

Scenario: description of a situation in the application domain.
Syntax: Title + Goal + Context + {Resources}$_1^N$ + {Actors}$_1^N$ + {Episodes}$_2^N$ + {Exceptions}

Title: identification of the scenario. In the case of a sub-scenario, the title is the same as the episode sentence (see below in the Episode definition), without the constraints.
Syntax: Phrase | ([Actor | Resource] + Verb + Predicate)

Goal: aim to be reached in the application domain. The scenario describes the achievement of the goal.
Syntax: [Actor | Resource] + Verb + Predicate

Context: composed by at least one of the following sub-components:
Geographical Location: physical set of the scenario.
Temporal Location: time specification for the scenario development.
Precondition: initial state of the scenario.
Syntax: {Geographical Location} + {Temporal Location} + {Precondition}
where Geographical Location is:
Phrase + {Constraint}
where Temporal Location is:
Phrase + {Constraint}
where Precondition is:
[Subject | Actor | Resource] + Verb + Predicate + {Constraint}

Resources: relevant physical elements or information that must be available in the scenario.
Syntax: Name + {Constraint}

Actors: persons, devices or organization structures that have a role in the scenario.
Syntax: Name

Episodes: set of actions that details the scenario and provides its behavior. An episode can also be described as a scenario.
Syntax (using partial BNF):
<episodes> ::= <group series> | <episode series>
<group series> ::= <group> <group> | <non-sequential group> | <group series> <group>
<group> ::= <sequential group> | < non-sequential group>
<sequential group> ::= <basic sentence> | <sequential group> <basic sentence>
<non-sequential group> ::= **#** <episode series> **#**
<episode series> ::= <basic sentence> <basic sentence> |
<episode series> <basic sentence>
<basic sentence> ::= <simple sentence> | <conditional sentence> | <optional sentence>
<simple sentence> ::= <episode sentence> CR
<conditional sentence> ::= **IF** <condition> **THEN** <episode sentence> CR
<optional sentence> ::= **[** <episode sentence> **]** CR
where <episode sentence> is described:
(([Actor | Resource] + Verb + Predicate) | ([Actor | Resource] + [Verb] + Title)) + {Constraint}

Exceptions: usually reflect the lack or malfunction of a necessary resource. An exception hinders the achievement of the scenario goal. The treatment of the exception may be expressed through another scenario.
Syntax: Cause **[(**Solution**)]**
where Cause is:
Phrase | ([Subject | Actor | Resource] + Verb + Predicate)
where Solution is:
Title

*Constraint***:** a scope or quality requirement referring to a given entity. It is an attribute of Resources, basic Episodes or sub-components of Context.
Syntax: ([Subject | Actor | Resource] + **Must [Not]** + Verb + Predicate) | Phrase

+ means composition, {x} means zero or more occurrences of x, () is used for grouping, | stands for **or** and [x] denotes that x is optional

vocabulary, in well-bounded situations and the benefits of the sub-scenario concept (a hierarchical relationship) improve the readability of our scenarios.

When there is a growth in the number of scenarios, the global perception of the UofD will decrease. We overcame this inconvenience by introducing integration scenarios, which collect several related situations in a larger scenario. Integration scenarios share with scenarios the representation and the links. Integration scenarios and scenarios are also bound in a hierarchical way.

2.1 Describing scenario components

Our scenario representation language [7](Fig. 1) is composed of static and dynamic components. The static components establish the settings of the scenario and are: title, goal, context, resources, actors, and constraints. The dynamic components express behavioural aspects through episodes and exceptions entities.

A scenario, univocally identified by its title, must satisfy a specific goal, occurs within a context and requires the availability of certain resources, and the participation of one or more actors. The context is described detailing a geographical location, a temporal location, and preconditions. The constraint attribute is used to characterize non-functional requirements applied to context, resources, and episodes.

Episodes are a set of actions that allow goal achievement. They are one of three types: simple, conditional, and optional. *Simple episodes* are those necessary to complete the scenario. *Conditional episodes* are those whose occurrence depends on a specified condition. *Optional episodes* are those that may or may not occur.

Independently of its type, an episode can be expressed as a single action or can itself be a scenario, thus allowing the possibility of *decomposition of a scenario in sub-scenarios*.

Our scenario representation provides the description of behaviour with different temporal orders. A sequence of episodes implies a *precedence order*, but a *non-sequential order* is provided by a special syntax (#<episode series>#), used to express a parallel or indistinct sequential order. A scenario may be interrupted by exceptions. Each exception is described using a simple sentence that specifies the cause of the interruption and where it may take place. If we include the title of another scenario, this one will treat the exception and may or may not satisfy the original goal.

Figure 2 exemplifies a scenario of the *Saving Plan for Automobile Acquisition System* [25]. This system will be used further throughout this article to exemplify the inspection process.

A seller of brand new vehicles offers long-term plans of payment in instalments. A group of customers is constituted to acquire a vehicle each paying monthly instalments. Every month, two vehicles are assigned to participants, one vehicle by drawing lots and the other by bidding. Drawing lots means that all participants have the chance to obtain a vehicle early by random assignment. Bidding means that those participants who want to get the vehicle immediately offer to pay several instalments in advance. The system manages the monthly vehicle assignment, the cashing of instalments, the constitution of groups, the substitution, abandon or cancellation of participants, the trace of debtors, and the recalculation of instalments in case of a factory change in vehicle models or prices. Therefore, this complex system requires many legal management and economical controls and enough flexibility to support constant changes in market and company policies.

2.2 Our scenario construction process

The general idea of our process, detailed in [7], is to anchor the scenario description on the vocabulary of the UofD. As such, we start from a pre-existing lexicon in order to build scenarios. The lexicon describes the application vocabulary and the set of scenarios describes the application.

The lexicon was first proposed in [26] and is called Language Extended Lexicon (LEL). It is a representation of the symbols in the application language, and it is anchored on a very simple idea: *understand the language of the problem without worrying about understanding the problem*. The Lexicon's goal is to register signs (words or phrases) that are peculiar to the domain. Each entry in the lexicon is identified by a sign or signs (in case of synonyms) and has two descriptions. The first, called *notion*, is the denotation (defines "what the symbol is") of the word or the phrase. The second, called *behavioural response* is the connotation (describes " how the symbols acts in the system") of the word or the phrase. Entries are classified into four types according to their general use in the UofD. The types are: subject, object, verb, and state.

Below, we exemplify an entry of LEL for a symbol based on the *Saving Plan for Automobile Acquisition System* (Fig. 3).

While describing the symbols, two principles are to be followed: the *principle of circularity*, also called the "principle of closure", and the *principle of minimal vocabulary* [26]. The first principle aims to maximize the use of signs in the description of other signs. The second one aims to minimize the use of symbols external to the lexicon. These rules stress the description of the vocabulary as a self-contained and highly connected hypertext [7].

The scenario construction process starts with the application domain lexicon. These scenarios are later improved using other sources of information and orga-

Title: **Build a group of adherents**
Goal: Build a new group of adherents for a saving plan.
Context:

Geographical Location: Administration society headquarters.
Temporal Location: Every second monday.
Precondition: There are as many accepted solicitors as required to build a group.

Resources: application form, **Constraint**: it must be numbered.
formal notification form.

Actors: administration society, accepted solicitor.
Episodes:

The administration society chooses among all accepted solicitors those who will integrate the new group.
The administration society generates a list of members of the new group of adherents including personnel data and number assigned to each application form.
The administration society records the new group of adherents to participate in the next adjudication act.
The administration society sends a formal notification of acceptance to every accepted solicitor included in the new group.

Exceptions: ---

Fig. 2 An example of a scenario

nized in order to obtain a consistent set of scenarios. During this process, the scenarios are verified and validated and discrepancies, errors, and omissions (DEO) are detected. Figure 4 [7], using an SADT model, describes the construction process. The SADT activities (boxes) are described below:

1. *Derive activity* consists of identifying actors, identifying candidate scenarios, and creating scenarios using LEL. These three steps are performed extracting information only from the lexicon by applying derivation heuristics.
2. *Describe activity* consists of completing the derived scenarios adding information from the UofD, from the previous lexicon elicitation process, and from the lexicon itself. During this activity, new scenarios might be discovered and the scenarios' content should be improved.
3. *Organize activity* consists in reorganizing scenarios, defining relations among scenarios, and finally grouping them together in integration scenarios. The reorganize step is based on the composition and decomposition of scenarios in order to improve scenario comprehension and management. Two or more scenarios are put together when a unique situation became artificially separated. A scenario is divided when it contains more than one situation. Scenario relationships are identified in order to be able to integrate scenarios; they are: hierarchical, overlap, order, and exception relationships. Integration scenarios are built based upon the detected relationships.

Fig. 3 An example of a lexicon symbol

Member of group
Notion:

- he or she is an accepted solicitor belonging to a group.

Behavioral Response:

- he or she can make a bidding.
- he or she participates in a drawing lots if the installments were paid in term.
- he or she may abandon the saving plan.
- he or she may cancel the debt.
- he or she may pay installments before due date.
- he or she pays installments in authorized banks.
- he or she gets the chosen vehicle once all the installments were paid.

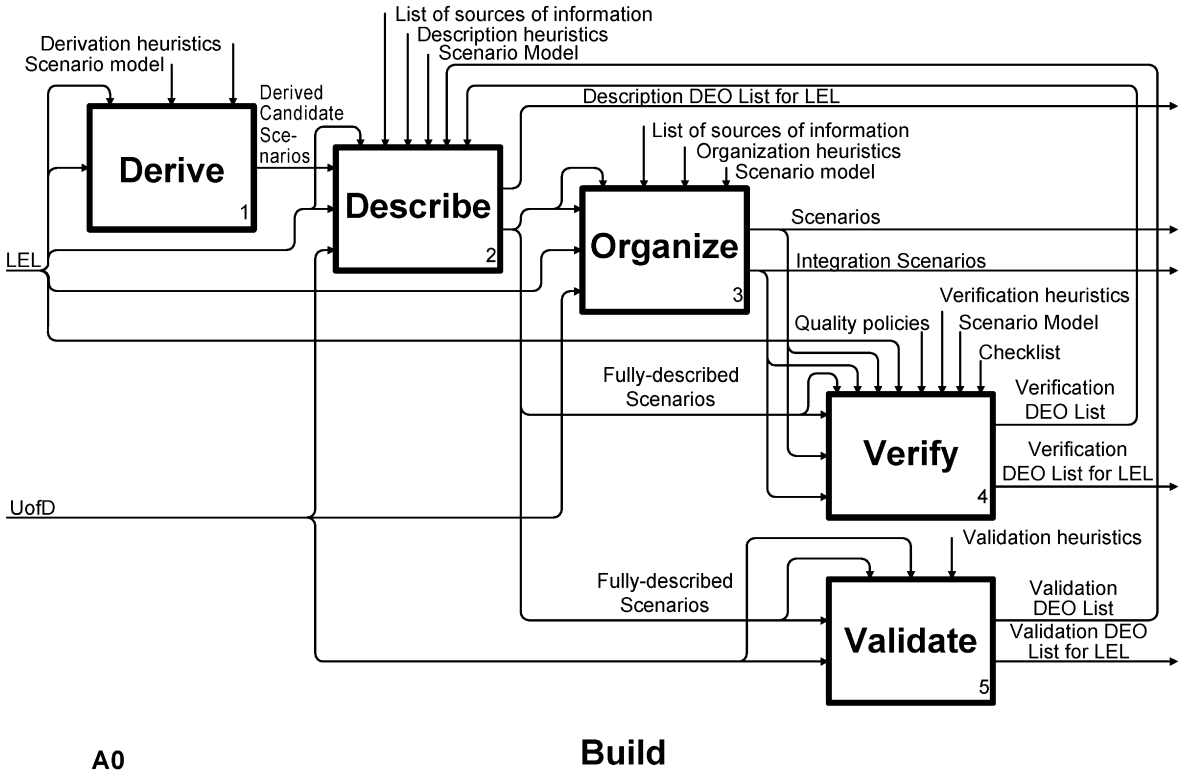


Fig. 4 SADT of the scenario building process (adapted from [7])

4. *Verify activity* aims to obtain consistent scenarios by defect detection and is performed by means of two different verification processes and an evaluate activity. Figure 5 shows, in detail, the verify activity whose main output is the verification DEO list. The intra-verify activity provides a systematized review of individual scenarios while the inter-verify activity takes into account the set of scenarios as a whole. The evaluate activity generates a final disposition report, which indicates final acceptance or the next step to be taken: a new verification once scenarios have been repaired. Scenario corrections are done as part of the describe activity based on the verification DEO list produced here.
5. *Validate activity* mainly aims to confirm elicited information and to detect defects and, as a side effect, to elicit new information. This activity is carried out performing interviews or meetings where clients and users are in charge of reading and discussing the scenarios.

The SADT depicted in Fig. 5 helps to clarify the idea of verification. It divides the verification into three parts; the first two are a consequence of the scenario model, that is, we can verify a given scenario or we can verify the relationship among scenarios. It can be seen that in both activities 1 and 2, the control arrows intra-verification heuristics and inter-verification heuristics are present. The fact that we have chosen inspections as our verification paradigm means that those control arrows will be defined by our inspection method, which is described in detail in Sect. 4.

3 Inspections

An ad-hoc reading of scenarios finds defects. However, if there are available tips about where defects may be located and how they can be perceived, the task of finding them becomes easier and reader productivity increases. Experienced readers find more defects than novice ones. In all areas of human knowledge, experience can be condensed into tools, methods, rules, or procedures to help inexperienced persons in complex tasks.

Reviews applying different procedures have been formalized. Among those, inspections have emerged as one of the most effective quality assurance techniques in software engineering [15, 16]. Fagan [12] was the inventor of software inspection as a formal process composed of six well-defined steps: *planning*, *overview*, *preparation*, *meeting*, *rework*, and *follow-up*.

The actual use of the inspection technique in the field led to the development of several variants of inspections. An example is the N-fold technique [27, 28], which uses multiple inspection teams. Schneider reported that his research pointed out that a nine-fold replication of the inspection process raised the fault-detection rate from 35% (single independent team) to 78%. Kantorowitz et al. [29] proposed a simple probabilistic model to predict the fault detection ratio based

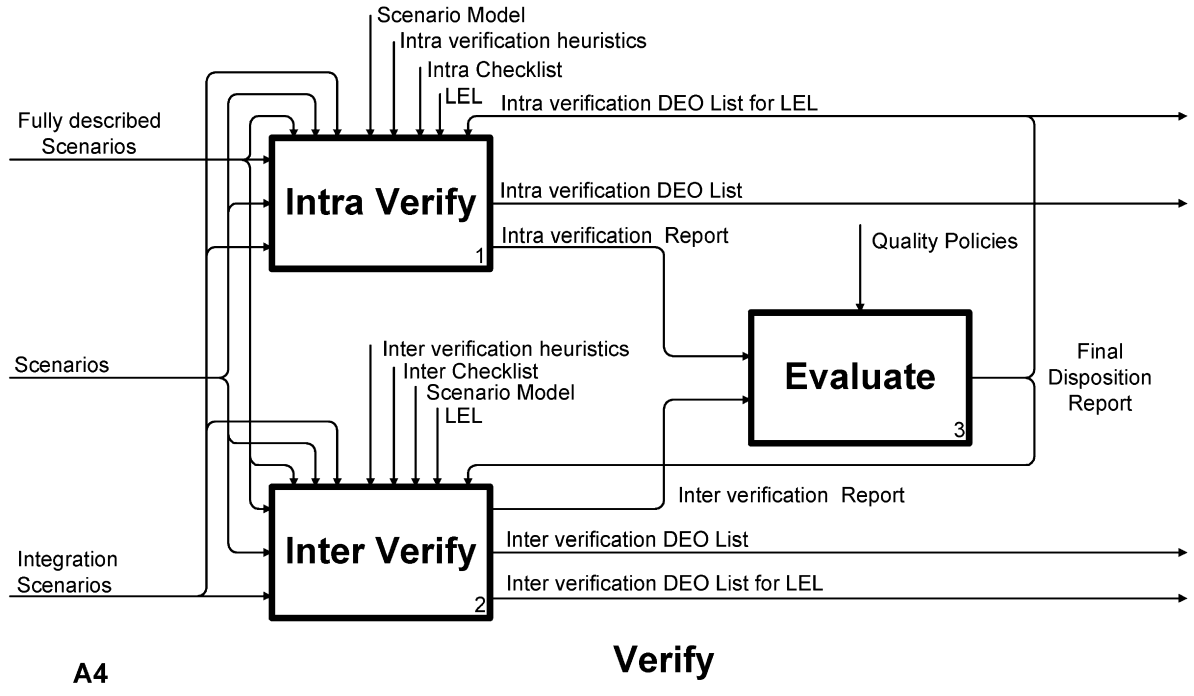


Fig. 5 SADT of the verify activity

on two variables: level of expertise of the inspectors and the number of inspection teams.

Reading techniques for the *preparation* step may be classified into four groups. They were classified in [17] as: ad hoc, checklist, scenario-based, and constructive reading. The checklist is the closest to Fagan's proposal. Figure 6 depicts the taxonomy of the preparation step reading approaches.

- *Ad hoc* reading is done in an unstructured fashion, whereby the inspector receives very little support. This does not mean that inspection participants do not scrutinize the artefact systematically. It only means that no previous plan is available and everything is left to the inspector's experience.
- *Checklist* reading gives the inspector strong support in the form of a list of questions and checks that have to be answered and verified in a certain pre-established order. The checklist approach has several weaknesses such as the lack of help in understanding the artefact and poor tailoring to a given development environment.
- *Scenario-based* reading moves the centre of the inspection from the meeting towards the preparation step. This technique provides the inspector with detailed guidance about how to find specific defects. Examples are: Basili [30] on perspective reading, Porter [19] on defect-based reading, and Cheng [31] on function point reading.
- The term *constructive reading* means that the inspection process goes further than merely reviewing an artefact and producing a list of defects. During constructive reading, the inspector produces a new representation of the inspected material, which is

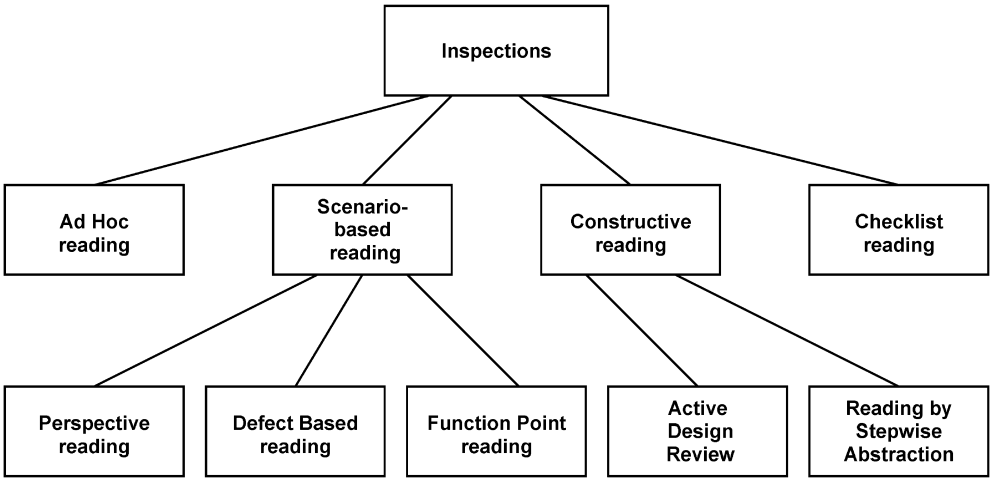


Fig. 6 Taxonomy of reading approaches

Table 1 Number of experiments comparing the performance of scenario-based with ad hoc and checklist reading

Scenario-based reading performance	Ad hoc	Checklist
Better than	10	5
Same as	4	5
Worse than	–	–

analysed later. Variants of this technique are: active design review [32] and stepwise abstraction [33].

Several experiments have been carried out comparing different inspection techniques applied to requirements documents: [19, 30, 34, 35, 36, 37, 38, 39]. Table 1 summarizes the findings of these experiments. Scenario-based inspections performed better than ad-hoc inspections in ten experiments and better than checklist in five experiments. We should keep in mind that not all experiments surveyed compared scenario-based with ad-hoc or checklist.

In some experiments, the experience of the inspector was followed more carefully [37] and the results showed that the importance of the procedure definition decreases when the inspector experience increases. One experiment [17] showed, in two cases, no difference between defect-based reading and perspective reading. The role played by the different participants involved in the inspection process also changes from one incarnation to another of the inspection notion. Actually, it seems that there are no changes in the effectiveness of the process due to variations in the specific task each person has been assigned.

The inspection process described throughout this paper is close to Porter's [19], since the main idea is to make use of the available knowledge about scenarios and their defects. In fact, the scenario structure may look naive at first sight; however, its structure, even when flexible, gives many anchors for consistency and completeness checks.

Defect-based reading requires a set of detailed heuristics reflecting the knowledge about what is being read. Our proposal concentrates all available knowledge about scenario properties in forms, so it could be called form-based reading. However, to keep it within the mainstream of the inspection literature it is called *scenario-based*. The first version of our inspection strategy was introduced in [40].

4 Our inspection strategy

The aim of the scenario inspection is to guarantee the best possible quality of the scenario set under production. The flexibility and simplicity of the scenario structure allows the existence of several defects such as:

- Missing information in scenarios
- Scenarios with erroneous information
- Scenarios with ambiguities
- Scenarios holding contradictions
- Partially or totally overlapping scenarios

Knowing the type of the defects beforehand improved the detection process. Since we deal with different types of defects, a brief taxonomy is given in Sect. 4.1.

Our inspection strategy is a way of conducting the verify activity in the scenario construction process (Figs. 4 and 5). Thus, its main purpose is to provide feedback to the process in order to improve its quality. The outputs of the verify activity are the DEO lists, which, in our case, will be produced by means of a defect-based inspection. Section 4.2 describes this strategy.

As mentioned before, our defect-based inspection is based on forms. Section 4.3 describes these forms in detail. Section 4.4 gives insight on the management viewpoint of our inspection process. It focusses on forms that we have used to collect data on the process itself. This information should empower moderators and help with the production of feedback information about the inspection process. Section 4.5 refers to the fact that, following Parnas [41] advice, we have presented our method with the improvements made after the case studies. Therefore, this section reports on how we designed our method using the feedback from the case studies.

4.1 Defects taxonomy

Many terms can be used in relation to the type of defects found during a review, for example, contradictions, discrepancies, errors, inconsistencies, ambiguities, omissions, and conflicts. They may be arranged in three groups:

1. Conflict, contradiction, and inconsistency share, in this field, their meaning with discrepancy, which can be outlined as "the presence of two or more elements showing different and incompatible issues". At least one of the issues is not true.
2. Error is simply a statement that is not true.
3. Omission involves missing facts. An ambiguity is a special kind of omission where the defect is seen as a minor lack of information that can be avoided choosing one of several possible interpretations.

We define

- **D** to be the set of all discrepancies
- **E** to be the set of all errors
- **O** to be the set of all omissions

What we call a defect throughout this article is $\mathbf{D} \cup \mathbf{E} \cup \mathbf{O}$. Since $\mathbf{D} \cap \mathbf{E} \neq \emptyset$, a part of a discrepancy is also an error. In the data collected from the case studies, they were counted once, as discrepancies. From now on, the terms *defects* and *DEOs* are used interchangeably throughout the text.

Besides the classification DEO for defect type, we also use another sorting criterion related to defect severity, which is composed of three types: fundamental, organizational, and presentation. Severity defects are detailed in Sect. 4.4.

4.2 Scenario inspection

Our inspection process relies on Fagan's proposal [12] with some carefully defined changes to better suit both our scenario construction process and the inspected product.

The overview phase has been deliberately removed from our proposal. In this phase, authors should present a global vision of their product. Since the set of documents the inspector receives (scenario set, LEL, forms and instructions) has to be self-explanatory, any incomprehension that arises at this point is an indication that either there are problems with the documents or the inspector failed to read them. Suppressing the overview phase has the extra advantage of keeping away from the author's bias. This initial independence between inspector and authors increases the relevance of meeting.

We also do not have the follow-up phase as a step in our inspection strategy since we deal with it in the context of our own scenario construction process. Figure 4 shows the feedback from the verify activity to the describe activity. Section 4.4 details some of the management issues related to the outputs of the inspection process.

The entire preparation phase is devoted to identify defects rather than obtaining knowledge of the inspected material and it is biased to a filling in the blanks in predesigned forms, since the inspector reads the document in order to complete the provided forms.

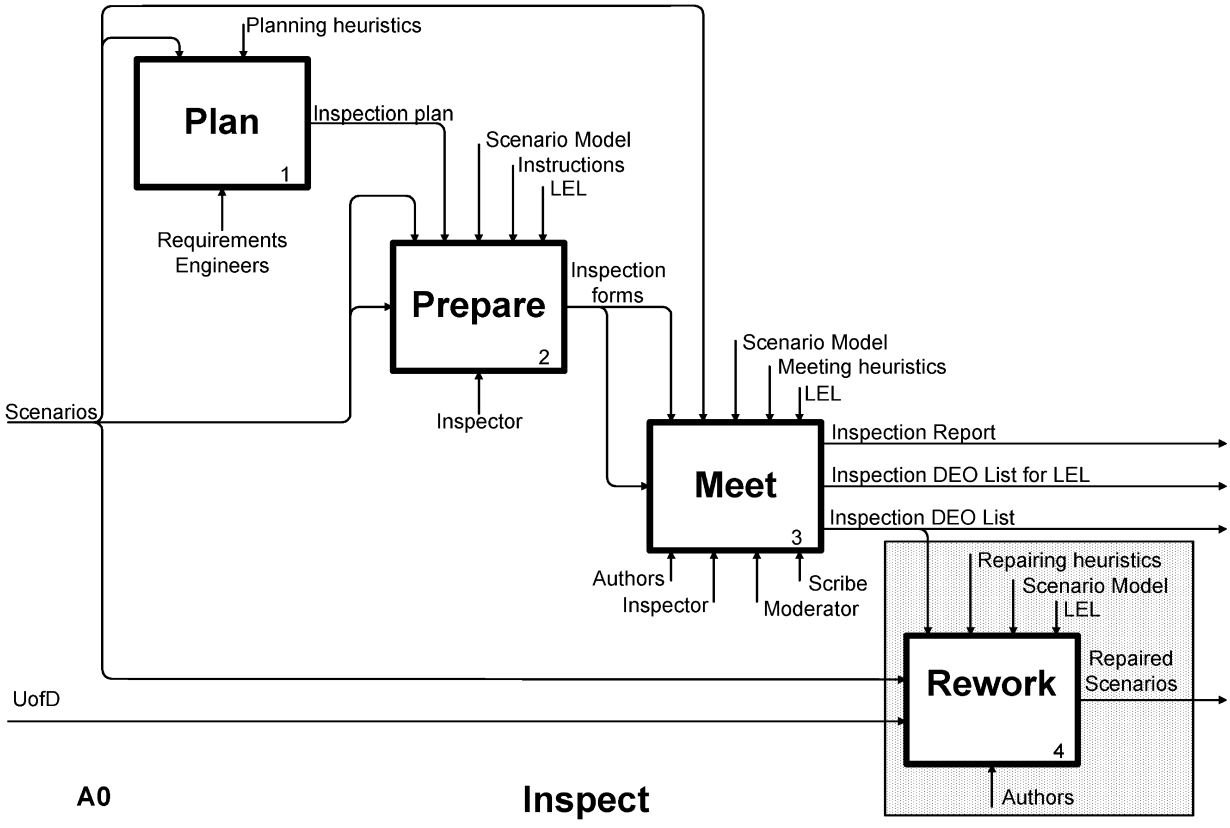
This activity schema is based on what Porter [19] calls scenario-based reading, since the inspector uses several systematic techniques to find different types of defects. Meeting aims are to validate the preparation-detected defects with the authors and to find new defects.

As it is shown in Fig. 7, our inspection process comprises four activities: planning, preparation, meet, and rework.

The planning activity consists of selecting the material to be inspected, selecting the participants, defining their roles (inspector, moderator, and scribe), and delivering the material to the inspector (scenario set, LEL, forms, and instructions). The requirements engineers produce an inspection plan containing the following data: name of participants and their roles, date when the material is delivered to the inspector, estimated preparation deadline, and estimated meeting date.

Preparation consists of reading to understand LEL and reading the instructions, followed by the filling in of

Fig. 7 SADT of our inspection process



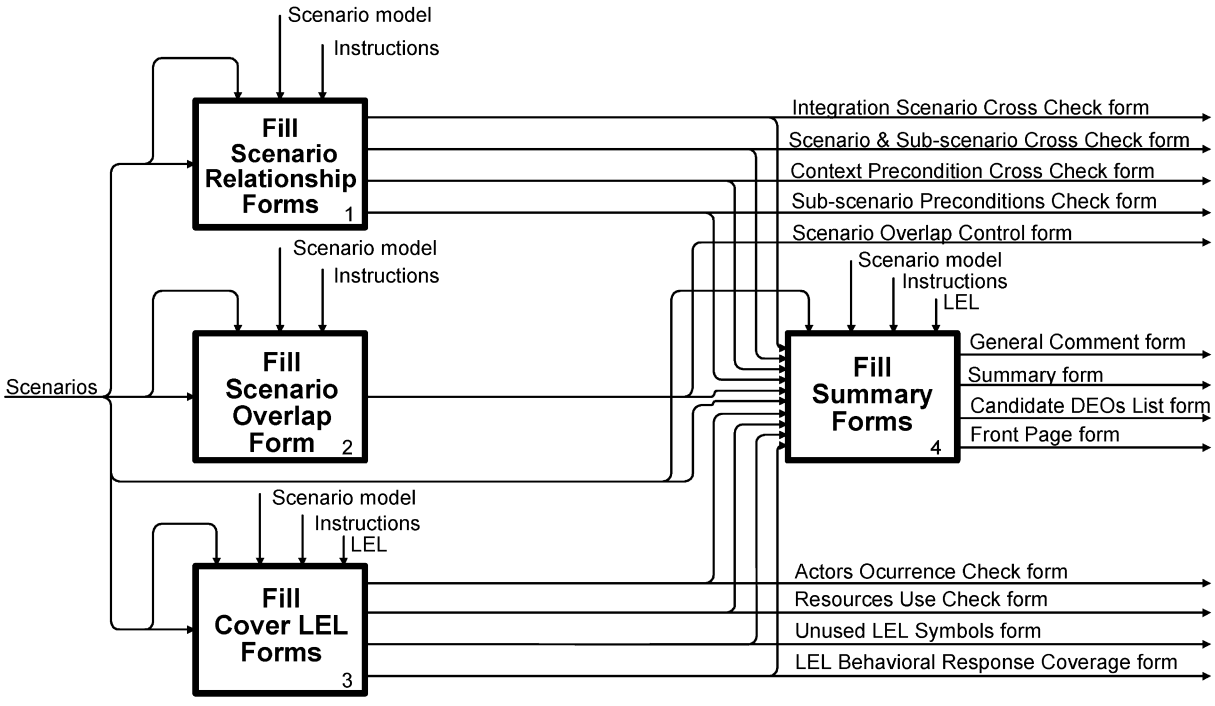


Fig. 8 SADT of the inter-scenario inspection prepare activity

forms, registering DEOs, and raising questions about doubts. Instructions contain general guides to help fill in the forms. For every form, instructions define the form objective and give detailed filling guides and defect analysis guides.

During the meeting activity, the inspector and the scenario authors discuss every preparation-detected point while the scribe takes notes of every conclusion reached. Each point confirmed as a defect or as an unanswered question is registered in the inspection DEO list. Eventually, during meeting, the inspector or the authors need to review other portions of the scenario set to support their point of view. In this way, they may come across other defects. The moderator drives meeting to ensure that every point is adequately treated. He/she produces an inspection report document with some advice regarding the final disposition of the scenario set.

Afterwards, authors carry out the rework activity. New and more precise information is elicited from the UofD when the questions are answered and an improved version of the scenario set is obtained. This activity is not done as part of the inspection process, but it is an integrated part of the overall feedback involving verify and describe activities as depicted in Fig. 4. Since we preferred to keep Fig. 7 as close to the Fagan model as possible, Figs. 4 and 7 show some degree of overlap, precisely in the rework activity.

4.2.1 Intra- and inter-scenario inspection

The scenario inspection strategy presented in this article is actually composed of three independent processes. They are intra-scenario inspection, inter-scenario inspection, and evaluation (see Fig. 5). Planning, preparation, meeting, and rework activities are carried out for both intra- and inter-scenario inspection. Evaluation defines the next step to be taken: a re-inspection after rework or final acceptance.

The intra-scenario inspection verifies each component in every scenario to confirm its consistency with other components and adherence to the scenario model. The inter-scenario inspection checks the relationship among different scenarios looking for overlaps or gaps.

The intra-scenario inspection is carried out at least once. When the severity of the defect included in the DEO list justifies it, a new intra-scenario inspection may be required after the rework activity. When the intra-scenario inspection is finished, the inter-scenario inspection is performed one or more times if necessary.

The *intra-scenario inspection* focusses on the analysis of each scenario using LEL and the scenario model as guides; once the process is finished, an improved scenario set is obtained.

As it was mentioned above, our inspection approach extensively uses pre-designed forms during the preparation phase. This is the most time-consuming and most effective activity to detect defects in the inspection. The

Table 2 Description of the intra-scenario inspection forms

Form	Description
I	Front page: identifies the project, the scenarios to be inspected, LEL version, the scenario set author or authors, and the inspector. Relevant dates, scenario construction effort, and preparation effort are also registered.
II	Scenario quantitative summary: gives a general overview of the information volume present in every inspected scenario. It holds the number of actors, resources, episodes, LEL symbols used, constraints, and exceptions among other figures.
III	Scenario syntactic check: aims to verify whether the scenario components, excluding episodes, are correctly written. The allowed syntax is displayed in the form.
IV	Episode syntactic check: aims to verify whether the episodes of a given scenario are correctly written. A row of the form is filled for each episode failing to accomplish the syntax. Columns to register the missing or surplus information are available for the main body of the episode and for the constraint if present.
V	Episode Ppragmatic Ccheck: is used to detect irrelevant episodes or episodes with omissions of actors or resources.
VI	Lexicon compliance: aims to detect incorrect use of LEL symbols in a given scenario. Two possible wrong uses are considered: a) when one symbol is used, whether it is emphasized or not, with a meaning different from the one registered in LEL, and b) when the symbol is correctly used without being identified as an LEL symbol.
VII	Scenario actors occurrence check: aims to verify the coherence among the persons and organizational structures that have a role in the scenario with the actual list included in the actors component. Candidate actors are checked to see if they are LEL symbols or not.
VIII	Scenario resources use check: is very similar to form VII since it helps to check the coherence of the passive elements, support media, and artefacts listed in the resources component against those used along the scenario. Their inclusion in LELis also checked.
IX	Episode types syntactic check: is designed to help the detection of defects in the use of the special characters (#, [,]) and keywords (if, then) needed to represent conditional and optional episodes or to group them to denote lack of temporal ordering (Fig. 1). It helps with checking the syntax of the scenario model (Fig. 1), in particular the episode grammar.[a]
X	Candidate DEOs list: is divided into two sections. The first one is designed to hold the corrections suggested because of the defects detected during the preparation activity, and the second one contains those items that require extra information from the UofD. In other words, DEOs are grouped into two sets based on the possibility of solving them using the available information. Every detected defect or question is tagged with the form number and page where it was discovered.
XI	General comment: gives a summary of the scenario quality based on the DEOs detected, the corrections proposed, and the unanswered doubts.

[a] If this was to be automated, a simple parser could detect most of these defects.

forms belonging to the intra-scenario inspection are grouped into four sets, as follows:

1. Syntactic verification
2. Relationship with LEL
3. Relationship among components
4. Inspection summary

Section 4.3.1 describes in detail the forms used in the intra-scenario inspection.

The *inter-scenario inspection* is performed when the scenario set has been organized (see Sect. 2.2) and the intra-scenario inspection finished; again, when this process is completed, an improved scenario set is obtained.

As it can be seen from the data presented in Sect. 5, the prepare activity of the inter-scenario inspection is the most important one. It also uses pre-designed forms, which are grouped into:

– Scenario relationship
– Scenario overlap
– LEL coverage
– Inspection summary

Sect. 4.3.2 describes in detail the forms used in the inter-scenario inspection.

Every form related to the activities 1–3 in Fig. 8 is filled in independently from the others. Summary forms are completed based on the previous ones. The actual defect detection is carried out in activities 1–3 while the last activity collects their findings.

4.3 Inspection forms

Every form has a complete set of attached instructions. They specify how the forms should be completed and which aspects of the scenarios should be analysed to capture defects.

During the filling in of any form, the inspector sometimes comes across defects whose detection is not the objective of the form itself. This sort of unplanned defect detection is called spontaneous detection and, according to our experience (see Sect. 5), accounts for about 15% of the total number of defects.

Title: Build a group of adherents

Goal: Build a new group of adherents for a saving plan.

Context:

Geographical Location:
Administration society headquarters

Temporal Location:
Every second monday.

Precondition:
There are as many accepted solicitors as required to build a group.

Resources:

application form **Constraint**: numbered formal notification form

Actors:

administration society, every accepted solicitor

Episodes:

The administration society chooses among all accepted solicitors those who will integrate the new group.

The administration society generates a list of members of the new group of adherents including personal data and number assigned to each application form.

The administration society records the new group of adherents to participate in the next adjudication act.

The administration society sends a formal notification of acceptance to every accepted solicitor included in the new group.

Exceptions: ---

Scenario under inspection

COMPONENT <SYNTAX>	MISSING	SURPLUS
Title <Phrase | Sentence>	---	---
Goal <Sentence>	---	---
Geographical Location <Name>	---	---
Temporal Location <Name>	---	---
Preconditions <Sentence>	---	---
Constraint <Coercion>	---	---
Actors <Name>		every
Resources <Name>		---
Constraint <Coercion>	must be	---
Exceptions <Cause [Title]>	---	---

Form III: Scenario Syntactic Check

SYMBOL	LEL	LACK OF IDENTIFICATION	WRONG USE
Administration society	Subject	---	---
Group of adherents	Subject	Goal, Episode 2	---
Application form	Object	---	---
Formal notification	Object	---	---
Accepted solicitor	Subject	---	---

Form VI: Lexicon Compliance

	OCURRENCE				EXISTANCE	
Candidate	**Title**	**Goal**	**Context**	**Episodes**	**Actors**	**LEL**
Group	1	1	1	1, 2, 3, 4	No	Subject
Administration society	0	0	0	1, 2, 3, 4	Yes	Subject
Accepted solicitor	0	0	1	1, 4	Yes	Subject

Form VII: Scenario Actors Occurrence Check

In form III, Sentence is <[[Subject] | Actor | Resource] + Verb + Predicate> and Coercion is <[Subject | Actor | Resource] + must [not] + Verb + Predicate>. Data for Title, Goal and Context in form VII mean number of uses.

Fig. 9 Examples of intra-inspection forms linked to a scenario

In the following sections, all inspection forms are presented together with some data taken from the case study, saving plan for automobile acquisition system.

4.3.1 Intra-scenario inspection forms

The intra-scenario inspection is simpler than the inter-scenario inspection but requires a greater effort. It is simpler since every verification step has a small scope and few issues to be concerned with. The effort required to fill in these forms is mainly related to the huge amount of small details to be taken into account.

The forms for syntactic verification (see forms III, IV, and IX in Table 2) check the syntax for all the components of a scenario. To make a scenario comply with LEL means to ensure that LEL symbols are properly used and that every phrase emphasized as a LEL symbol is actually part of LEL (see form VI). Every subject in an episode must be listed in the actors component and any actor must play a role in an episode. This is checked in the relationship among components step (see form VII). Resources and episodes require a similar check. A semantic-biased relationship among components is the one existing between the goal and episodes components. The set of episodes must fulfil the scenario goal. This is also verified using the relationship among component forms (see forms V and VIII). The inspection summary forms (see forms I,

Form VI. LEXICON COMPLIANCE

Objective:

Detect incorrect use of lexicon symbols, such as the use of a symbol with a meaning different from the one recorded in LEL, the omission of the link to the LEL and others.

Steps:

- Record every word or phrase identified as a lexicon symbol in the column SYMBOL, disregarding synonyms and repeated uses.
- For each presumable symbol, check its presence in LEL and record its type in the column LEL. When the potential symbol does not belong to LEL, register this fact in the form X.
- For every confirmed symbol: read carefully the scenario looking for unidentified uses of the symbol; register the scenario components where this occurs in the column LACK OF IDENTIFICATION and also record the fact in the form X.
- For every confirmed symbol: read carefully the scenario looking for uses of the symbol with a meaning different from the one described in LEL; register the scenario components where this occurs in the column WRONG USE and in the form X too.

Analysis:

When a lexicon symbol has been used in the scenario without being identified as such but its use matches the meaning recorded in LEL, the correction of the defect simply consists in producing a link to LEL.
When the use of a confirmed symbol is not compatible with the meaning recorded in LEL, the correction of the defect consists in replacing the mention of the symbol in the scenario component by a synonym not included in LEL.

Fig. 10 Instructions for the intra-inspection form VI: lexicon

Table 3 Description of the inter-scenario inspection forms

Form	Description
I	Front page: provides the same information as the one used in the intra-scenario inspection.
II	Scenario quantitative summary: is very similar to the one used in the intra-scenario inspection. It gives a general overview of the information volume present in the scenario set, such as number of scenarios by type, number of actors, resources, episodes, LEL symbol references, constraints, and exceptions among other figures.
III	Integration scenario cross check: is used to verify that those scenarios tagged as integration scenarios are correctly classified and described.
IV	Scenario and sub-scenario cross check: allows verification of the existence of sub-scenarios mentioned in episodes or exceptions and, conversely, the correct classification of sub-scenarios never mentioned by other scenarios.
V	Context precondition cross check: aims to detect errors and discrepancies in scenario preconditions and omissions of information in other scenarios that would satisfy those preconditions.
VI	Sub-scenario preconditions check: aims to detect errors in sub-scenario preconditions or omitted data in those scenarios that mention sub-scenarios, taking into account the hierarchical links established between scenarios and sub-scenarios and between integration scenarios and scenarios.
VII	Scenario overlap control: is used to detect the overlap of information among scenarios. To reduce comparisons among scenarios, a scenario proximity index is calculated based on the coincidence of actors, resources and context; when a high index is obtained, comparisons between goals and episodes are performed.
VIII	Actors occurrence check: aims to detect actors not included as LEL symbols but having great participation in scenarios.
IX	Resources use check: aims to detect resources not included as LEL symbols but whose availability is required in many scenarios.
X	Unused LEL symbols: is used to detect omissions in scenarios by checking the level of coverage of LEL.
XI	LEL behavioural response coverage check: aims to detect omitted situations and discrepancies between behavioural responses of LEL subjects and scenarios/episodes involving the corresponding actors.
XII	Candidate DEOs list: provides the same kind of information as the one used in the intra-scenario inspection, so it is also divided into two sections. The first one is designed to hold the corrections suggested because of the defects detected during the inspection, and the second one contains those items that require extra information from the UofD. In other words, DEOs are grouped intp two sets based on the possibility of solving them using the information available. Every detected defect is tagged with the form and page number where it was discovered.
XIII	General comment: gives a summary of the scenario set quality, based on detected DEOs, suggested corrections, and emerged doubts.

II, X, and XI) collect all detected defects and unanswered questions as a guide to scenario update. A new knowledge elicitation from the UofD occurs during the rework activity.

A brief description of each intra-scenario inspection form is given in Table 2 while Fig. 9 shows examples of three of these forms. Figure 10 exemplifies the instructions attached to the intra-inspection form VI.

Form VI in Fig. 9 lists the lexicon symbols and their role in the lexicon (subject, object, verb, or state) together with a reference to which scenario component has the problem. For instance, "group of adherents" (Fig. 9)

SCENARIO	SUB-SCENARIO	PRECONDITION	RELATION
8	*9*	*The adherent must have monthly payments up-to-date.*	*Episode 2*
11	*12*	*There are as many adhesion applications as members in the group.*	*Episodes 1 and 2*

Fig. 11 Form VI: sub-scenario preconditions check

Fig. 12 Form VII: scenario overlap control

S_i	Sj	ACTOR		RESOURCE		CONTEXT		PROXIMITY INDEX I_{ij}	GOAL	EPISODE	OVERLAP
		$A_{\cup ij}$	$A_{\cap ij}$	$R_{\cup ij}$	$R_{\cap ij}$	$C_{\cup ij}$	$C_{\cap ij}$				
1	*4*	*2*	*1*	*3*	*0*	*2*	*0*	*---*			
1	*5*	*2*	*1*	*3*	*0*	*2*	*0*	*---*			
1	*6*	*1*	*1*	*3*	*1*	*2*	*0*	*0.33*	*0*	*0*	*---*
...	*...*	*...*	*...*	*...*	*...*	*...*	*...*	*...*	*...*	*...*	*...*
1	*11*	*2*	*1*	*4*	*0*	*2*	*0*	*---*			
1	*12*	*2*	*1*	*2*	*2*	*2*	*0*	*0.50*	*1*	*0*	*---*
1	*13*	*3*	*0*	*3*	*0*	*2*	*0*	*---*			

$$\mathbf{I_{ij} = (\alpha * C \cap_{ij} + \beta * A \cap_{ij} + \gamma * R \cap_{ij}) / (\alpha * C \cup_{ij} + \beta * A \cup_{ij} + \gamma * R \cup_{ij})}$$

Where: α , β , γ , are weight factors,

$C \cap_{ij} = |\, Ctx(S_i) \cap Ctx(S_j) \,|$, $A \cap_{ij} = |\, Act(S_i) \cap Act(S_j) \,|$, $R \cap_{ij} = |\, Res(S_i) \cap Res(S_j) \,|$,

$C \cup_{ij} = |\, Ctx(S_i) \cup Ctx(S_j) \,|$, $A \cup_{ij} = |\, Act(S_i) \cup Act(S_j) \,|$, $R \cup_{ij} = |\, Res(S_i) \cup Res(S_j) \,|$,

$Ctx(S_k)$: Context of Scenario k, $Act(S_k)$: Actors of Scenario k, $Res(S_k)$: Resources of Scenario k.

$\mathbf{I_{i,j}}$: is the proximity index of Scenarios i and j.

Fig. 13 Form XI: LEL behavioural response coverage check

SUBJECT	BEHAVIORAL RESPONSE	SCENARIOS - EPISODES	ACTORS
Adherent	*Adherent pays the monthly installment to the Administration society.*	*Scenario 13, Episode 1*	*Adherent or Adjudicator*
Adherent	*Adherent gives saving plan rights to a third one according to cession rules.*	---	---

is used in the scenario goal and in the second episode, but it is not underlined as it should be.

Form VII in Fig. 9 lists: words understood as actors in the scenarios, where these words appear, and if they are properly listed in the component actors of the scenario model. The last column of the form indicates whether the actor is in the lexicon and its role.

4.3.2 Inter-scenario inspection forms

The inter-scenario inspection deals with more complex issues than intra-scenario inspection since it handles all scenarios at the same time.

Hierarchical relationships among scenarios and sub-scenarios are verified using scenarios relationship forms (see forms III, IV, V, and VI in Table 3). Scenario overlap form (see form VII) deals with obscure and poorly defined borders among scenarios that share common portions. Gaps are the most difficult defects and they are partially detected in LEL coverage forms (see forms VIII, IX, X, and XI) since those symbols, when never used or used with low frequency, are indication of possible gaps among scenarios. The inspection summary forms (see forms I, II, XII, and XIII) collect all relevant data from previous forms.

Table 3 contains a short description of each inter-scenario inspection form. Figures 11, 12, and 13 show examples of these forms with data extracted from the saving plan for automobile acquisition system. Figure 14 exemplifies the instructions attached to the inter-inspection form VI, which is depicted in Fig. 11.

Figure 12 depicts an example of form VII, which aims to calculate a matching index between any two scenarios. If the index is higher than 0.5, then there is an indication that scenarios need to be compared in more detail. That is, it compares goals and episodes to determine whether there is an overlap or not. Columns S_i and S_j in Fig. 12 are filled with pairs of scenarios. For each pair the union of actors ($A_{\cup ij}$), the number of

Fig. 14 Instructions for the inter-inspection form VI: sub-scenario preconditions check

Form VI. SUB-SCENARIO PRECONDITIONS CHECK.

Objective:

Detect errors in the preconditions or omissions in scenarios containing sub-scenarios. This is done by using the hierarchical links of scenarios with sub-scenarios and integration scenarios with sub-scenarios.

Steps:

- Build a list including all pairs of scenarios linked with a sub-scenario. If the sub-scenario has preconditions, register them in the column PRECONDITION using one row for each precondition. If there are no preconditions, register the fact in the form XII.
- In the same way, add to the list all pairs of integration scenarios linked with scenario.
- Determine the relationship between each sub-scenario precondition and scenario precondition or scenario episodes located before the sub-scenario mention, and register it in the column RELATION. If no relation can be established, register the fact in the form XII adding any other extra useful information.

Analysis:

When the sub-scenario does not have any precondition, consider its possible omission. Two possible situations may occur:

a) Sub-scenario is not the first episode of the scenario.
b) Sub-scenario is the first episode of the scenario.

If the sub-scenario is not the first episode of the scenario, review the episodes preceding the sub-scenario. This sequence may define part of the sub-scenario preconditions. If the sub-scenario is mentioned as the first episode of the scenario, the scenario precondition may be sub-scenario precondition too.

When the precondition of the sub-scenario is not satisfied by a scenario episode and it does not coincide with the scenario precondition, three possible situations may take place.

a) An error or omission in the sub-scenario precondition.
b) An error or omission in the precondition of the scenario.
c) An omission of episodes in the scenario.

When case a) occurs, suggest the review of the sub-scenario precondition. If b), suggest the review of the scenario precondition. Upon the occurrence of case c), suggest considering the possibility of missing episodes or even scenarios. When none of these cases occurs, a general review of the scenario and sub-scenario must be done.

Table 4 Description of the management forms

Form	Description
I	Front page: identifies the project, the scenario set, LEL version, the scenario set author or authors, the inspector, the moderator, and the scribe. Relevant dates, amount of information obtained after preparation, scenario construction effort, preparation effort, and meeting effort are also registered. The final disposition of the scenario set is put down here.
II	DEOs summary: shows a summary of the forms. It contains three tables where the defects are organized by type (discrepancy, error, or omission), by severity, by component, and by way of detection. An observation cell allows the moderator to take down notes about the meeting and the scenario set quality.
III	Inspection process evaluation: organizes defects by severity, by detection mechanism, and by inspection form. Defects are also classified in two categories: those ones whose repair can be done by the scenario author/s and those that require going back to the UofD. The form is mainly used to evaluate the design of the inspection process.
IV	DEOs by component: shows defect totals by component, by type of defect, by type of detection, and by severity. The form is mainly used to evaluate the scenario defects found.
V	Rejected DEOs by form: deals with inspection defects, initially identified as scenario defects at preparation but dismissed in the meeting. The form shows the rejected defects by source of generation (the process or the inspector) and by form.

actor coincidences ($A_{\cap ij}$), the union of resources ($R_{\cup ij}$), the number of resource coincidences ($R_{\cap ij}$), the union of context statements ($C_{\cup ij}$), and the number of context statement coincidences ($C_{\cup ij}$) are calculated. The proximity index is calculated in order to reduce the searching space of scenario pairs to be analysed for overlapping.

4.4 Managing the inspection process

Meeting is the control point in the management of the inspection process. During the meeting, the participants develop a common view of the inspected material. Each DEO is examined, the reader's approach is evaluated, and the overall quality of the scenario set is brought to bear.

Fig. 15 Form IV: DEOs by component

	DISCREPANCIES		*ERRORS*		*OMISSIONS*	
Component	**SPONTANEOUS**	**PROCESS**	**SPONTANEOUS**	**PROCESS**	**SPONTANEOUS**	**PROCESS**
Title				7, 9		10
Goal				2		
Context						
Resources				4, 4		
Actors				4, 9, 10		
Episodes			9	4, 11, 12, 12, 13		10
Exceptions						
Constraints						
General						

Table 5 Description of case studies

Case study	Set of scenarios	
	Name	Number of authors
1	MSCH-1	2
2	SPAAS-1	1
3	SPAAS-2	3
4	SPAAS-3	4
5	SPAAS-4	4
6	MSCH-2	2
7	MSCH-1, v.2	2
8	CPTPPWS	1
9	CPTPPWS	1

Table 7 Total number of defects

Case study	Discrepancies	Errors	Omissions	Total DEOs
1	0	7	8	15
2	4	121	2	127
3	10	15	40	65
4	10	33	165	208
5	4	5	11	20
6	13	7	16	36
7	0	1	6	7
8	0	107	36	143
9	0	18	12	30

Two results are obtained as a consequence of meeting. The first one is the information required to decide about the final disposition of the scenario set, meaning that it may need rework and it may eventually require new inspection. The second one is a set of forms, called management forms, filled in with information to be used as a process feedback. These forms are described in Table 4.

The management forms are filled in based on the information from candidate DEOs list form. During meeting, scenario authors and readers get together with the scribe and the moderator to ratify or dismiss candidate DEOs. New defects may also be found in the process. Every piece of information collected during the preparation phase is reviewed, organized, and evaluated.

A severity level is assigned to every defect, taking into account the following possible values: fundamental, organizational, and presentation.

1. *Fundamental DEOs* are the ones related to the overall semantics of a scenario, for example, lack of a subscenario, scenario overlap, lack of compatibility among title and goal, lexicon behavioural responses not covered by any scenario, and lack of preconditions.
2. *Organizational DEOs* are the ones related to problems with syntax and structure such as: episode splitting, episode factoring, lack of supporting actors or resources, lack of a scenario component, actors/resources that appear in the episodes but are not presented in the actors/resources component.

Table 6 Case studies sizes

Case study	Type of inspection	Number of scenarios	Number of episodes	Number of filled forms	Number of filled lines	Preparation time (h)	Meeting time (h)
1	Intra	13	56	104	1,002	9 1/2	1 1/2
2	Intra	19	92	171	1,135	7 1/2	–
3	Intra	12	44	84	585	5 1/2	–
4	Intra	17	76	181	1,234	15	–
5	Intra	16	51	95	529	9	–
6	Intra	17	63	170	849	12 1/2	2
7	Inter	13	63	25	299	6	1
8	Intra	45	235	380	1,519	11 1/2	2
9	Inter	45	235	60	691	14	1 1/2

Table 8 Total number of defects at preparation and meeting

Case study	Discrepancies		Errors		Omissions		Total DEOs	
	Prepare	Meeting	Prepare	Meeting	Prepare	Meeting	Prepare	Meeting
1	0	2	7	17	8	3	15	22
6	13	0	7	23	16	8	36	31
7	0	0	1	1	6	5	7	6
8	0	0	106	1	35	1	141	2
9	0	0	16	2	12	0	28	2

Table 9 Number of defects detected during preparation classified by type of detection

Case study	Process detection	Spontaneous detection	% Process/ total	% Spontaneous/ total
1	15	0	100%	0%
2	121	6	95%	5%
3	52	13	80%	20%
4	179	29	86%	14%
5	20	0	100%	0%
6	15	21	42%	58%
7	7	0	100%	0%
8	141	7	95%	5%
9	28	1	97%	3%

Table 10 Number of defects detected during meeting by type of detection

Case study	Process detection	Spontaneous detection	% Process/ total	% Spontaneous/ total
1	17	5	77%	23%
6	12	19	39%	61%
7	6	0	100%	0%
8	2	0	100%	0%
9	2	0	100%	0%

Table 11 Number of defects detected at preparation by severity

Case study	Fundamental DEOs	Organizational DEOs	Presentation DEOs
1	0	4	11
2	3	122	2
3	5	46	14
4	9	111	88
5	5	13	2
6	1	35	0
7	2	0	5
8	1	41	99
9	1	23	4
Total	27	395	225

Table 12 Number of defects detected at meeting by severity

Case study	Fundamental DEOs	Organizational DEOs	Presentation DEOs
1	1	17	4
6	0	16	15
7	2	3	1
8	1	1	0
9	0	2	0
Total	4	39	20

3. DEOs which neither affect the comprehension nor the consistency of the scenario are labelled as *presentation DEOs*, for instance, due to badly written sentences or LEL symbols used but not marked. Spelling and grammar mistakes were not considered as defects.

The moderator indicates the defect severity and how it was detected while the scribe registers this information in the forms. If there is any doubt, it is discussed with the other meeting participants. The moderator resolves possible conflicts. This approach allows for evaluating the inspection process, inspector efficiency, and the overall quality of the inspected scenarios.

The management forms described in Table 4 are applied to both the intra- and inter-inspection processes. Figure 15 presents an example of form IV with organizational defects of case study 1 (see Table 5) picked up at meeting. The numbers in the form represent the scenario numbers where defects were found.

4.5 Observations about the inspection process design

The construction of the inspection process went over five phases.

1. Design of the inspection forms
2. Redaction of the instructions
3. Preliminary use of forms and instructions
4. Design and fill in of management forms
5. Full application of the inspection process

Earlier steps of the inspection design process required more and deeper changes than later steps. Three researchers with experience in building LEL and scenarios worked during the design of the inspection forms. The design of forms was centred on the difficulties found while building scenarios. In addition, the detection of other defects to ensure scenario consistency was later included.

The goal of the instructions were to be self-contained and easy to use. Usage of the instructions should be such

Table 13 Number of defects detected at preparation by scenario component[a]

Case study	Title	Goal	Context	Resources	Actors	Episodes	Exceptions	Constraints	General
1	2	3	0	1	5	4	0	0	0
2	1	3	1	42	15	33	3	8	21
3	7	1	10	25	18	4	0	0	0
4	0	11	18	46	15	55	0	0	63
5	0	3	0	2	5	4	0	2	4
6	1	1	2	19	5	8	0	0	0
7	0	0	0	0	0	3	0	0	4
8	11	2	43	15	3	66	0	0	0
9	1	0	15	0	0	6	0	0	7
Total	23	24	89	150	66	182	3	10	99

a It is worth explaining why the component exception is not much used. We understand that most of the writers were not using a modelling strategy where exceptions become explicit. However, some scenarios did use the "if then" statement of the scenario model (Fig. 1), which in part explains this situation. Overall we have stressed in our scenario writing instruction the importance of exceptions. On the other hand, exceptions are generally detected under validation, and here we are performing just verification.

Table 14 Number of defects detected at meeting by scenario component

Case study	Title	Goal	Context	Resources	Actors	Episodes	Exceptions	Constraints	General
1	4	3	1	2	3	9	0	0	0
6	1	0	3	13	2	12	0	0	0
7	0	0	0	0	0	2	0	0	4
8	1	0	0	1	0	0	0	0	0
9	0	0	1	0	0	1	0	0	0
Total	6	3	5	16	5	24	0	0	4

that neither previous knowledge of the scenarios to be inspected nor experience on the inspection process itself were necessary. The instruction guides contain a description of each form with the following structure: the form objective, the steps to complete the form, and the analysis to be performed. Figures 10 and 14 exemplify those instructions.

Afterwards, a preliminary use of forms and instructions was performed over four case studies. This step allowed the improvement of the forms and of the instruction guides. Changes mainly referred to details that helped in the understanding of the instructions.

Once four candidate DEO list forms were filled in, the necessity of new forms to organize the information already produced became evident. Thus, the design and fill of management forms step was performed.

After forms and guides were corrected we performed the full application of the inspection. The intra-scenario inspection was applied to two case studies and the inter-scenario inspection was performed once.

A different inspector performed each case study. Forms and instructions were delivered to inspectors and they were advised to avoid asking about the scenarios and the inspection process (plan activity). After reading carefully the instruction guides they started to fill in forms (prepare activity). Once forms were completed, a meeting was established among scenario authors, inspector, scribe, and moderator for each case study. During the meeting activity the candidate DEO list was analysed and, when necessary, the involved scenario was read to clarify the DEO found. Some defects recorded in the candidate DEO list were rejected since they were the inspector's errors and other defects were confirmed, assigning the corrections to authors.

During form completion, inspectors recorded comments about inspection and those considered improvements to the process where taken into account. We observed that people without knowledge of the UofD and of the scenarios could fill in the forms but the knowledge of the scenario representation was a requirement. Inspections performed by people without the scenario representation knowledge presented problems because of the inspector's errors. These problems were perceived during meeting. So, although a reading technique is a plus, as per Sect. 3, we conclude that meeting should not be overlooked. Our observations agree with Fagan's regarding the critical aspect of meeting.

As mentioned in the opening of Sect. 4, we have followed a learning process in building the method. The forms were changed when necessary, from the feedback of the process, adding new ways for the detection of defects and getting rid of form components that were not important.

5 Data from case studies

In order to obtain information from our strategy we conducted nine case studies overall. Each case study is

defined by applying the intra- or inter-scenario inspection process to a set of scenarios. There were six different sets of scenarios produced by different people[2]. In [7] we detailed how each of these sets was constructed. Table 5 gives an overall description of each case study.

In Table 5, each set of scenarios is identified by a unique name. A different group for a given application produced each set. We used three applications:

MSCH Meeting scheduler system [42], the case study proposed by van Lamsweerde [43] applied to Universidad de Belgrano. The goal of the system is to manage the scheduling of meetings for a given organization.

SPAAS Saving plan for automobile acquisition system [25]. The goal of this system is to manage saving plans for the acquisition of brand new vehicles. A group of physical or legal persons is constituted and, monthly, participates in an adjudication process organized by a general manager in order to deliver a vehicle.

CPTPPWS Ceramic Paving Tiles Production Plant warehouse system. The goal of this system is to manage the inventory of Ceramic Paving Tiles Production Plant including machinery spare parts. The system handles suppliers and internal customers.

Since we applied both intra- and inter-scenario inspection to the set of scenarios MSCH-1, we generated the MSCH-1v.2, which is the set of scenarios MSCH-1 with changes due to the intra-inspection process (the rework activity, Fig. 7).

The case studies involved ten different kinds of people, including the authors of MSCH-1, MSCH-2, and CPTPPWS. Five people were faculty members and five were students. The number of scenarios subject to the inspection process was 195, 22 being the average number of scenarios in a set of scenarios. Table 6 describes each case study regarding effort, measured in hours, in preparation as well as the time used during meeting.

The process requires the filling in of a special form with information about DEOs. This is done in two stages of the process: at the end of preparation and at the end of meeting (see Fig. 5). The data for the total DEOs for each case study are given in Table 7. Following, Table 8 shows the DEOs detected at both stages.

From Tables 7 and 8 we have an overall idea of the improvement in quality made possible by applying scenario inspections. Table 8 shows a comparison of DEOs found after preparation and after the inspection meeting. DEOs found at the preparation stage are carried through to meeting, where some DEOs may be dismissed or new DEOs could be included.

We also measured, by asking the inspectors, whether the DEOs were found due to the process or were found by a side effect of applying the process. We have called the latter case a spontaneous detection in opposition to process detection. Tables 9 and 10 show the data gathered for the Preparation and Meeting stages. It is important to point out that more than 80% of the DEOs were found by the detection process.

Case study 6 was the only one where the inspector was not familiar with the scenario representation and loosely followed the instructions spending too much time, compared to the others, in reading the set of scenarios before applying the process. Even though there is no confirmation of this, it strongly suggests that the knowledge of the scenario model is an unavoidable requirement to become an efficient reader.

Severity of DEOs was ranked in three levels: fundamental, organizational, and presentation, as described in Sect. 4.4.

Tables 11 and 12 provide severity data for each case study. The classification of a DEO into a severity level is performed by the inspector and registered in the management forms. Most of the DEOs detected during the intra-scenario inspection process are of an organizational level. The only case study that performed the inter-inspection process was case 7. We noticed here that the fundamental severity was predominant; this was expected since taking into account relationships among scenarios gives more opportunity to find fundamental DEOs.

We also gathered the DEOs by scenario component (see Fig. 1) at preparation and at meeting. Tables 13 and 14 present the data. Most of the DEOs (65%) did occur in the resources, actors, and episodes components. General DEOs are the ones not directly limited to a specific component. We believe that the data displayed is a solid argument for using an inspection process as a verify activity in the process of scenario construction. In general, the effort for the inspection process is 25% of the total effort time for the scenario construction. We firmly believe that it is a reasonable overhead for the quality improvement achieved. It is also important to stress that if automated tools were available, the total effort time will be greatly reduced because of the amount of effort necessary to fill in the forms (Table 6).

6 Conclusions

This article presents a strategy for scenario inspections to be performed by requirements engineers as a verification process before validation with clients. This strategy was designed to be integrated with a specific construction process [7] and was applied in nine case studies. The data collected soundly support our claim that scenario inspections do improve scenario quality. We have also reported elsewhere [20], in detail, that our proposed strategy can be used, with some adaptations, in other processes and representations based on scenarios.

[2]Graduate and undergraduate students performed most of the writing of the scenarios, but case 1 and 7 were performed by software professionals.

Although scenario inspection is not a new idea [18], our proposal is based on an inspection strategy not employed before. Our reading of Gough's process is that an ad-hoc approach to preparation was used. Based on experimentation data available [17, 19, 29, 30, 34, 35, 36, 37, 38, 39], we believe that our proposal, a scenario-based one, is more effective.

The lack of the overview phase joint with the self-explanatory characteristic of all documents involved (scenario set, LEL, forms, and instructions) avoid initial bias and enhance the importance of the meeting. People without knowledge of the UofD and of the scenarios to inspect could fill in the forms but they had to be familiar with the scenario representation.

We see our contributions being useful not just for researchers but for practitioners, as well. For practitioners, we believe that our work, which has matured over the years with several case studies, provides yet additional convincing arguments that a verification process can and should be done as early as possible in the software construction process. We also understand that our processes, based on forms, can be adapted to different scenario production processes in use in different organizations. For researchers, we understand that our model does advance the state of the art with respect to applying verification concepts of inconsistency to natural language-based descriptions as well as integrate an extended lexicon to anchor semantic aspects of the descriptions.

The drawbacks of our proposal, too schematic and labour intensive, can be softened by the use of intelligent editors and verification agents. However, we should not forget that the main effectiveness factor of applying inspection methods is that a human being is in control of the process. The techniques and tools will help, but they will have a supporting role.

References

1. Leite JCSP, Rossi G, Balaguer F, Maiorana V, Kaplan G, Hadad G, Oliveros A (1997) Enhancing a requirements baseline with scenarios. Req Eng J 2(4):184–198
2. Rolland C, Ben Achour C, Cauvet C, Ralyté J, Sutcliffe A, Maiden M, Jarke M, Haumer P, Pohl K, Dubois E, Heymans P (1998) A proposal for a scenario classification framework. Req Eng J 3(1):23–47
3. Jacobson I, Christerson M, Jonsson P, Overgaard G (1992) Object-oriented software engineering—a use case driven approach. Addison Wesley, Reading, Mass., ACM Press, New York
4. Wirfs-Brock R (1995) Designing objects and their interactions: a brief look at responsibility-driven design. In: Carroll J (ed) Scenario-based design: envisioning work and technology in system development. Wiley, New York, pp 337–360
5. Rolland C, Ben Achour C (1998) Guiding the construction of textual use case specifications. Data Knowledge Eng 25:125–160
6. Weidenhaupt K, Pohl K, Jarke M, Haumer P (1998) Scenarios in system development: current practice. IEEE Softw 15(2):34–45
7. Leite JCSP, Hadad GDS, Doorn JH, Kaplan GN (2000) A scenario construction process. Req Eng J 5(1):38–61
8. Ross D, Schoman A (1977) Structured analysis for requirements definition. IEEE Trans Softw Eng (Special issue on requirements analysis) 3(1):6–15
9. Gilb T (1987) The principles of software-engineering management. Addison-Wesley, Reading, Mass.
10. Yourdon E (1989) Structured walkthroughs, 4th edn. Prentice Hall, New York
11. Parnas DL, Weiss D (1985) Active design reviews: principles and practices. In: Proceedings of the 8th international conference on software engineering, pp 132–136
12. Fagan ME (1976) Design and code inspections to reduce errors in program development. IBM Syst J 15(3):182–211
13. Fagan ME (1986) Advances in software inspections. IEEE Trans Softw Eng 12(7):744–751
14. Ackerman AF, Buchwald LS, Lewsky FH (1993) Software inspections: an effective verification process. IEEE Softw 6(3):31–36
15. Gilb T, Graham D (1993) Software inspection. Addison-Wesley, Reading, Mass.
16. Laitenberger O, DeBaud JM (2000) An encompassing life-cycle centric survey of software inspection. J Syst Softw 50(1):5–31
17. Regnell B, Runesom P, Thelin T (1999) Are the perspectives really different? Further experimentation on scenario-based reading of requirements. Requirements engineering with use cases – a basis for software development. Technical Report 132, Paper V, Department of Communication Systems, Lund University, pp 141–180
18. Gough PA, Fodemski FT, Higgins SA, Ray SJ (1995) Scenarios—an industrial case study and hypermedia enhancements. In: RE95: proceedings of the international symposium on requirements engineering. IEEE Computer Society Press, Los Alamitos, Calif., pp 10–17
19. Porter AA, Votta LG Jr, Basili VR (1995) Comparing detection methods for software requirements inspections: a replicated experiment. IEEE Trans Softw Eng 21(6):563–575
20. Leite JCSP, Doorn JH, Hadad GDS, Kaplan GN (2003) Using scenario inspections on different scenarios representations. Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, N33/03
21. Zorman L (1995) Requirements envisaging by utilizing scenarios (rebus). PhD dissertation, University of Southern California
22. Carroll J (1995) Introduction: the scenario perspective on system development. In: Carroll J (ed) Scenario-based design: envisioning work and technology in system development. Wiley, New York, pp 1–18
23. Booch G (1991) Object oriented design with applications. Benjamin Cumming, Redwood City
24. Potts C (1995) Using schematic scenarios to understand user needs. In: Proceedings of DIS'95 - symposium on designing interactive systems: processes, practices and techniques. ACM Press, University of Michigan, pp 247–256
25. Mauco V, Ridao M, del Fresno M, Rivero L, Doorn J (1997) Ingeniería de requisitos, proyecto: sistema de planes de ahorro. Technical Report, ISISTAN, UNCPBA, Tandil, Argentina
26. Leite JCSP, Franco APM (1990) O uso de hipertexto na elicitaçao de linguagens da aplicaçao. In: Anais de IV Simpósio Brasilero de Engenharia de Software. SBC, Brazil, pp 134–149
27. Martin J, Tsai WT (1990) N-fold inspection: a requirements analysis technique. Commun ACM 33(2):225–232
28. Schneider G, Martin J, Tsai WT (1992) An experimental study of fault detection in user requirements documents. ACM Trans Softw Eng Method 1(2):188–204
29. Kantorowitz E, Guttman A, Arzi L (1997) The performance of the N-fold inspection method. Req Eng J 2:152–164
30. Basili V, Green S, Laitenberger O, Lanubile F, Shull F, Sorumgard S, Zelkowitz M (1996) The empirical investigation of perspective-based reading. J Empirical Softw Eng 2(1):133–164
31. Cheng B, Jeffrey R (1996) Comparing inspection strategies for software requirements specifications. In: Proceedings of the 1996 Australian software engineering conference, pp 203–211
32. Parnas DL (1987) Active design reviews: principles and practice. J Syst Softw 7:259–265

33. Dyer M (1992) Verification-based inspection. In: Proceedings of the 26th annual Hawaii international conference on system sciences, pp 418–427
34. Ciolkowski M, Differding C, Laitenberger O, Münch J (1997) Empirical investigation of perspective-based reading: a replicated experiment. ISERN report no. 97-13
35. Fusaro P, Lanubile F, Visaggio G (1997) A replicated experiment to assess requirements inspection techniques. Empirical Softw Eng 2(1):30–57
36. Miller J, Wood M, Roper M (1998) Further experiences with scenarios and checklists. Empirical Softw Eng 3(1):37–64
37. Porter AA, Votta LG Jr (1998) Comparing detection methods for software requirements inspections: a replication using professional subjects. Empirical Softw Eng 3(4):355–380
38. Sandall K, Blomkvist O, Karlsson J, Krysander C, Lindvall M, Ohlsson N (1998) An extended replication of an experiment for assessing methods for software requirements. Empirical Softw Eng 3(4):381–406
39. Shull F (1998) Developing techniques for using software documents: a series of empirical studies. PhD thesis, Computer Science Department, University of Maryland
40. Doorn J, Kaplan G, Hadad G, Leite JCSP (1998) Inspección de Escenarios. In: Proceedings of WER'98, workshop en engenharia do requisitos, Maringá, Brazil, pp 57–69
41. Parnas DL, Clements PC (1996) A rational design process: how and why fake it. IEEE Trans Softw Eng 12(2):251–257
42. Hadad G, Kaplan G, Leite JCSP (1998) Léxico extendido del lenguaje y escenarios del meeting scheduler. Technical Report no. 13, Departamento de Investigación, Universidad de Belgrano, Buenos Aires
43. van Lamsweerde A, Darimont R, Massonet P (1993)The meeting scheduler system—preliminary definition. Internal Report, Université Catholique de Louvain