Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Evidence of software inspection on feature specification for software product lines

Iuri Santos Souza[a,c,*], Gecynalda Soares da Silva Gomes[b], Paulo Anselmo da Mota Silveira Neto[a,c], Ivan do Carmo Machado[b,c], Eduardo Santana de Almeida[b,c,d], Silvio Romero de Lemos Meira[a]

[a] *Federal University of Pernambuco (UFPE) – Recife, Pernambuco, Brazil*
[b] *Federal University of Bahia (UFBA) – Salvador, Bahia, Brazil*
[c] *Reuse in Software Engineering Group (RiSE), Brazil*
[d] *Fraunhofer Project Center (FPC) for Software and Systems Engineering, Brazil*

ARTICLE INFO

Article history:
Received 21 December 2011
Received in revised form
20 November 2012
Accepted 23 November 2012
Available online 20 December 2012

Keywords:
Software quality control
Software inspection
Software product lines
Empirical study

ABSTRACT

In software product lines (SPL), scoping is a phase responsible for capturing, specifying and modeling features, and also their constraints, interactions and variations. The feature specification task, performed in this phase, is usually based on natural language, which may lead to lack of clarity, non-conformities and defects. Consequently, scoping analysts may introduce ambiguity, inconsistency, omissions and non-conformities. In this sense, this paper aims at gathering evidence about the effects of applying an inspection approach to feature specification for SPL. Data from a SPL reengineering project were analyzed in this work and the analysis indicated that the correction activity demanded more effort. Also, Pareto's principle showed that incompleteness and ambiguity reported higher non-conformity occurrences. Finally, the Poisson regression analysis showed that sub-domain risk information can be a good indicator for prioritization of sub-domains in the inspection activity.

1. Introduction

Software product line engineering, with its inherent reuse of software artifacts, in combination with the usual iterative development, brings a set of improvements for the development activities, such as cost reduction, time-to-market improvement, and so on (Clements and Northrop, 2001). However, even with these benefits, a set of challenges for quality assurance, a very important activity for SPL, arise (Denger and Kolb, 2006).

Thus, in order to ensure that the artifacts produced are in conformance with the required quality levels, techniques such as testing and software inspection can be applied. While testing is the dynamic quality assurance technique most commonly applied on executable artifacts (Tian, 2001), software inspection is a static quality assurance technique which can be performed on each software artifact created in the software development life-cycle (e.g. requirements, design, test cases, code, and so on) (Tian, 2001). According to Wiegers (2002), software inspections are among the most efficient software quality assurance techniques, especially in earlier life-cycle phases, since the sooner the non-conformities are detected it is less costly their correction (Tian, 2001). Software inspections can lead to the detection and correction of anywhere between 50% and 90% of defects (Wiegers, 2002; Gilb and Graham, 1993). Moreover, inspection brings also important education and social benefits. Young software developers can more rapidly learn standards for specification and code while working as inspectors, while expert developers under pressure are more tempted to ignore standards (Pezze and Young, 2007).

In addition, another important aspect to be considered in the SPL development is the variability management. Variability management techniques can be assessed by quality assurance techniques, such as inspection, to increase the reliability of the artifacts produced to be reused. Although some authors (Wiegers, 2002; Gilb and Graham, 1993; Boehm, 1987) advocate the importance of software inspections in every life-cycle phase, few studies describe inspection activities in the SPL context, as discussed in Denger and Kolb (2006), and highlighted by Babar et al. (2010). It can indicate a lack of substantial literature detailing quality assurance techniques which deals with variability in software engineering.

Based on this scenario, this study aims to understand how inspection should be handled in SPL scenario, to gather evidence of inspection on scoping artifacts, as well as, identify possible gaps that have not been addressed by current research. This study was conducted on the basis of an empirical study, aimed at investigating inspection-related issues during the application of a systematic

* Corresponding author. Tel.: +55 71 92531132.
E-mail addresses: iurisin@gmail.com (I.S. Souza), gecynalda@yahoo.com (G.S. da Silva Gomes), pamsn@cin.ufpe.br (P.A. da Mota Silveira Neto), ivanmachado@dcc.ufba.br (I. do Carmo Machado), esa@dcc.ufba.br (E.S. de Almeida), srlm@cin.ufpe.br (S.R. de Lemos Meira).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.11.044

software inspection approach in a SPL project. It also discusses issues such as the effort to perform inspections in the project, the most common non-conformities and the lessons learned.

The remainder of this paper is organized as follows: Section 2 presents the related work. Sections 3 and 4 detail the context and the design of the empirical study carried out on this work. Section 5 presents the research questions. Section 6 presents the proposed inspection approach. Sections 7 and 8 present the results, the analysis and the main findings. Section 9 reports the lessons learned. Section 10 presents the drawbacks and validity threats of the study and finally, Section 11 describes the conclusions and future directions.

2. Related work

Despite its importance, inspection of feature specifications and SPL artifacts have not been substantially addressed by researchers (Babar et al., 2010). As consequence, there are not many related studies in the literature.

Next, we describe three studies related to understand and evaluate the role of inspection in feature specifications.

Maßen et al. (2004) discussed some types of deficiencies that a feature model might contain. The authors claim that many deficiencies result from the overlapping semantic between domain relationships and dependencies. They classified the deficiencies in redundancies, anomalies and inconsistency. They also suggested solutions for solving some cases of deficiencies, grouped into some subcategories of these three types.

In Felty and Namojoshi (2003), the authors present an approach to automatically detect conflicts on feature specifications. Their work presented a method for detecting feature conflicts as features are specified as a collection of temporal logic formulas or automata. Interactions are discovered by finding pairs of specification formulas that are contradictory with respect to axioms about the system behavior.

Another study uses metamorphic testing for the automated generation of test data for the analyses of feature models (Segura et al., 2011). The authors define metamorphic relations between feature models and their set of products and a test data generator relying on them. The authors implemented also a prototype tool to automate it. The tool receives as input the feature model and its associated set of products, applies random transformations on the feature model and returns a set of neighbor models together with their associated set of products. The solution also performs automatic inspection to extract the expected outputs from a number of analyses over the generated models.

As previously stated (Babar et al., 2010), there is a lack of approaches and evidence about inspection activities on feature specifications, specially, in the industrial setting. Hence, this work can be seen as an initial effort toward understanding how a SPL project can benefit from applying inspection practices during feature specification activities.

3. The SPLSmart project and the empirical study context

Based on the lack of substantial literature discussing on quality assurance and techniques from variability point of view (Babar et al., 2010), particularly in the SPL context, this paper presents an empirical study performed in order to understand and characterize how inspection should be handled in the SPL scoping phase and their artifacts. The empirical study was performed within an industrial software project, which is composed of four products. This section presents the SPLSmart project overview and the empirical study context.

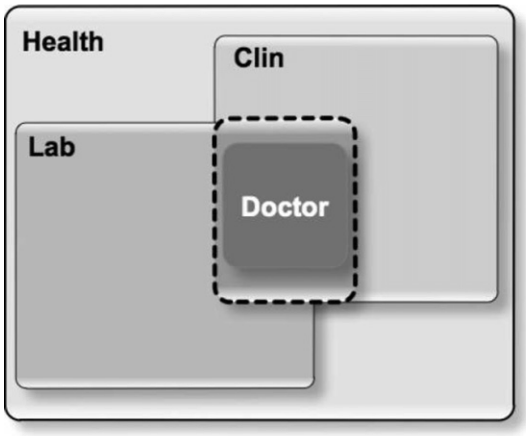


Fig. 1. Company products and their relationships.

3.1. Company background

The studied company has been working on the medical and health information systems domains since 1994. The company has more than 50 customers and a total of 51 staff members, distributed in different areas.

The company offers strategic and operational solutions for hospitals, clinics, labs and private doctor offices. The company's product portfolio includes a set of 42 modules, which are responsible for specific functions in different sub-domains (e.g. financial, inventory control and so on). Depending on the product that is being assembled, different sub-domains can be joined. The four products are the following:

- *SmartDoctor:* a web-based product composed of 11 sub-domains. Its goal is to manage the tasks and routines of a doctor's office.
- *SmartClin:* a desktop-based product composed of 28 sub-domains. It performs clinical management support activities, e.g. medical exams, diagnostics and so on.
- *SmartLab:* a desktop-based product composed of 28 sub-domains. It integrates a set of features to manage clinical pathology labs.
- *SmartHealth:* a desktop-based product composed of 35 sub-domains. It manages the whole area of a hospital, from financial to patient issues.

Fig. 1 shows the relationship among the different products, as well as their intersections.

Market trends, technical constraints and competitiveness motivated the company to migrate their products from single-system development to a SPL approach. Before starting the SPL adoption, the company developed its products independently, most of them from scratch, applying some form of ad-hoc reuse. In this context, the RiSE Labs[1] has devoted efforts toward introducing SPL practices in such company.

Firstly the SPL scoping process (Balbino et al., 2011) was applied, followed by the requirements engineering process (Neiva et al., 2010). From the SPL project scoping phase, a *product map* containing 4 products and more than 800 features were built.

3.2. Technology transfer

In a partnership with the RiSE Labs, the organization started the SPL adoption project. RiSE Labs has developed a new approach for product lines engineering called RiPLE which is composed of

[1] http://labs.rise.com.br.

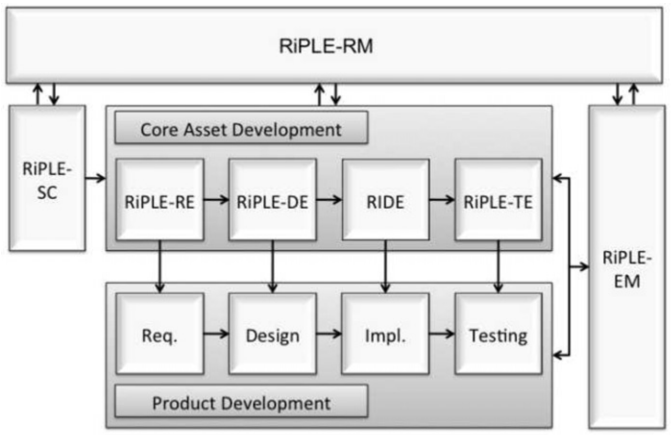


Fig. 2. RiPLE process.

disciplines for scoping (Balbino et al., 2011), requirements (Neiva et al., 2010), design (Filho et al., 2008; Cavalcanti et al., 2011a), risk management (Lobato et al., 2010), testing (Anselmo et al., 2010; Machado et al., 2011), evolution (Anastasopoulos et al., 2009), and so on. The cooperation was attractive for both partners: the company had no research department that could tailor a SPL approach to the company specific goals, and RiSE Labs could have a validation partner for its new approaches. This way, RiSE Labs conducted the technology transfer of SPL engineering techniques in the company. It involved the integration of the RiSE Labs team with the company staff in order to establish a set of activities toward the institutionalization of the product line.

Some RiPLE process disciplines were applied to the SPL development at the company. Fig. 2 shows the RiPLE process, where the boxes represent the disciplines covered by the entire process. The following RiPLE disciplines were applied in this study:

- **RiPLE-SC:** This discipline is an agile and systematic process for scoping, responsible for identifying the SPL potential (Balbino et al., 2011).
- **RiPLE-RE:** It is composed of the requirement engineering activities related to the refinement of the SPL scoping, the definition of requirements and use cases with variability issues. They are composed of common tasks, such as the elicitation, specification and verification, which are performed in an iterative and systematic way (Neiva et al., 2010).
- **RiPLE-DE:** It focuses on defining a Domain Specific Software Architecture (DSSA), which represents the architectural elements from the software product line. Such architectural definition must enable variability among products in a certain domain and must take advantage of commonalities among those products in order to promote software reuse (de Oliveira Cavalcanti et al., 2011).
- **RiDE:** It defines a systematic process to perform domain engineering, which includes the steps of domain analysis, domain design, and domain implementation (de Almeida, 2007).
- **RiPLE-TE:** It provides a structured process for testing in SPL projects, in which different testing phases are reported, such as: unit testing, integration, system and acceptance testing (Machado et al., 2011; da Mota Silveira Neto, 2010).
- **RiPLE-RM:** It is responsible for risk identification, documentation, analysis, and monitoring of risks elicited during the process execution (Lobato et al., 2010).
- **RiPLE-EM:** It is a systematic way to guide and manage the evolution of every asset and product in a product line context, handling change management, build management and release management activities (Burgos, 2008).

The project started with a training on SPL and SPL scoping, with the presentation of the RiPLE-SC discipline. Other workshops were also performed during the project, addressing different SPL aspects and RiPLE disciplines.

The domain and market expert built a list of products previously developed by the company. It was the input for RiPLE-SC (Balbino et al., 2011). Based on domains with better market potential, products and features were identified for the product line. As output, a product map composed of the products and their respective features was generated. Next, the RiPLE-RE (Neiva et al., 2010) was instantiated, refining the product map and defining the requirements and use cases for the domain. As the main result, these activities generated a set of assets, which consider the SPL variability to support other RiPLE disciplines and enable product instantiation.

The RiPLE-RM (Lobato et al., 2010) was executed in parallel with the other RiPLE disciplines. During RiPLE-RM, the risks and their characteristics such as description, type, status, mitigation strategy and contingency plan were identified. As risk management is a continuous activity, it was necessary to keep track of the history of identified risks. Thus, the risk likelihood and impact were documented according to its occurrence, which can happen in different moments throughout the project development.

4. Understanding inspection on SPL features specification in an industrial setting

This study investigated the non-conformities found and the effort spent during the application of a systematic software inspection approach, in a reengineering project toward the introduction of SPL practices in a Brazilian company. More specifically, the study consisted of understanding the role of inspection practices in the feature specification artifacts in a SPL project. It also aimed at understanding the issues that arise with the effort required to perform inspections in the project. For example, it tried to identify the most common non-conformity types, the most problematic items of the specification templates and so on.

4.1. Definition of the empirical study

In the definition phase, the foundation of the study is determined. Its goals and research questions are defined. Thus, the following structure was defined:

Goal: The objective of this study was to understand and characterize the software inspection activity on feature specification artifacts, aiming to gather evidence on the required effort and non-conformities that may appear.

RiSE Labs members reviewed the first version of the protocol for the study. In order to ensure alignment, coherence and coverage of the study goal, and avoid bias, the research questions were also reviewed by the RiSE members and SPL experts. The main improvements were related to the structure and bias prevention on the research questions. It is important to highlight that these experts did not participate in the research.

4.2. Data collection procedure

Data collected in an empirical study can be quantitative or qualitative. Quantitative data involves numbers and classes, while qualitative data involves words, descriptions, pictures, diagrams, and so on (Runeson and Höst, 2009). In this sense, in this empirical study, we used a combination of qualitative and quantitative data to provide a better understanding of the studied phenomenon (Seaman, 1999).

For the empirical study, we applied archival and observation methods to collect data and information, what makes possible the triangulation of methods and greater convergence of research topics (Runeson and Höst, 2009). All collected data were treated as strictly confidential, in order to assure anonymity to all study participants, related to misuse of the information, and ensure the complete freedom during data collection procedures. This way, they were free to register any information or opinion without constraints. Furthermore, the subjects were free to talk about the benefits, drawbacks and difficulties regarding to the software inspection approach performed on the SPL artifacts within the study.

4.2.1. *Archival data*

Archival data refers to any document available in the organization, such as documents from different development phases, organizational charts, financial records, and previously collected measurements in the company (Runeson and Höst, 2009). For this study, we used archival data to collect information regarding non-conformities captured on software inspection activity, as well as the effort data recorded along with the inspection activity.

4.2.2. *Observation*

The data collection from observation technique enables the researchers to develop activities within the company and become part of the company staff. During some interactions, the data was collected in a systematic and unobtrusive way, allowing the capture of firsthand information about behavior and interactions (Shull et al., 2007). On the study, we used observations in all inspection meetings, combined with audio recording (Wallace et al., 2002), where meeting attendants interact with each other, generating information about the studied object. Based on the classification of approaches to observations defined by Runeson and Höst (2009), the observation was performed with (i) high awareness of the subjects being observed; (ii) low degree of interaction by the researcher.

The observations were recorded in notes and/or audio, capturing the main information and activities directed by aspects of the study. The notes were registered in real time. Six participants were observed at the company site during the inspection activity and a total of more than 26 h of audio were recorded in this step.

4.3. *Analysis procedure*

In this phase, the strategy involves the quantitative and qualitative analysis of the collected data. For quantitative data, our analyses includes descriptive statistics, economics analyses (Pareto's principle) (Gallegati, 2008), quality control analyses (Montgomery and Runger, 2006), correlation analyses (Gautheir, 2001), and the development of predictive models (Khoshgoftaar et al., 2001). The objective of using qualitative analysis is to derive conclusions from the data, keeping a clear chain of evidence (Yin, 2008).

Moreover, the relevant data from documents, assets and extracted statements, as well as the observations, were grouped and stored in the study database, in order to maximize the benefits from sources of evidence in the study (Yin, 2008).

4.4. *Validity procedure*

In order to reduce the threats to validity, countermeasures were taken in the study design and during the whole study. All the countermeasures followed the quality criteria in terms of construct, external and internal validity as discussed in (Yin, 2008).

Construct validity: two strategies were used:

- **Prolonged involvement:** In this strategy, the researchers had a long involvement with the object of study allowing to acquire tacit knowledge which enabled us to avoid misunderstandings and misinterpretations (Karlström and Runeson, 2006).
- **Peer debriefing:** It recommends that the analysis and conclusions are shared and reviewed by other researchers (Karlström and Runeson, 2006). It was achieved by conducting the analysis with two researchers, performing discussion in groups in which analysis and conclusions were discussed and supervised by a statistical researcher.

Table 1
Feature specification template.

ID:
Name:
Description:
Priority:
Binding time:
Feature type:
Parent feature:
Child feature:
Required feature:
Exclusionary feature:
Sub-domain:

Internal validity: For example, in this study, the participants could be afraid to express their real impressions, since they feel restricted by the audio records. It is a threat to internal validity and was mitigated by ensuring the anonymity on the meetings and completing the inspection instruments.

External validity: Although the study was conducted in a single organization, our intention was to enable analytical generalization, so that the results can be extended to cases with similar characteristics and hence for which the findings are relevant, i.e. they can define processes or a theory.

Reliability: This requirement was achieved by using two tactics: a detailed empirical study protocol; and a structured study database with all relevant and raw data such as meeting tapes, transcripts, documents, outline of statistical models and so on.

5. Research questions

The following research questions were defined in this study:

5.1. *RQ1. What is the distribution of non-conformities in the project?*

This question intended to characterize the distribution of non-conformities by sub-domain, classifying the occurrences according to different types of non-conformities and distribution of the non-conformities per items of feature template. Table 1 shows the template for feature specification. The feature specification document is composed of eleven fields: (i) ID; (ii) Name; (iii) Description; (iv) Priority; (v) Binding time; (vi) Feature type (Optional, Mandatory and Alternative); (vii) Parent feature; (viii) Child feature; (ix) Required feature; (x) Exclusionary feature; and (xi Sub-domain.

The data sample used in this analysis is composed of data from sub-domains, features and identified non-conformities. The sub-questions are detailed next.

5.1.1. *RQ1.1. How are the non-conformities spread in terms of sub-domains?*

This question aims at analyzing the quality of the feature specifications, considering the ratio of non-conformities amount found per number of features, for every Sub-domain. From this ratio, we evaluated the quality and non-conformity behavior of the studied sample Sub-domains. Moreover, this representation provides the baseline input for establishing the quality control program on future iterations.

5.1.2. RQ1.2. What is the distribution of non-conformity occurrences on types of non-conformities?

This subquestion characterizes the distribution of non-conformity occurrences on the types of non-conformities. Thus, it is possible to analyze if some non-conformity types present more occurrences than others. Such information may provide inputs to a defect causal analysis (Card, 1998) and project managers can better apply their efforts in order to propose strategies to mitigate it.

The non-conformities are classified within nine types, as defined by van Lamsweerde (2009), which proposes a classification for non-conformities based on requirement specification. As the literature does not present a similar classification related to feature specification, we believe that this more general definition could be used in this context. They are next described:

- *Incompleteness or Omission:* Absence or omission of information necessary to specify the domain and sub-domain features – e.g. the feature document template contains incomplete or partially complete items and entries.
- *Ambiguity:* Presence of a specification item that allows more than one interpretation or understanding – e.g. ambiguous term or statement.
- *Incorrectness or Inadequacy:* Presence of a specification item that is incorrectly or inappropriately described – e.g. feature specifications that not justify its presence in products of the SPL.
- *Inconsistency or contradiction:* A situation in which some specified feature contain constraints, priority or composition rules that are in conflict with other features and/or work products, such as requirements or use case documents.
- *Non-traceability or opacity:* Presence of features that do not specify or wrongly specify its identifier or interaction with other features – e.g. one feature that do not have a unique identifier or one child feature that does not specify its respective parent feature.
- *Incomprehensibility or unintelligibility:* A situation in which a specification item is stated in such a way that it is incomprehensible to the target stakeholders.
- *Non-Organization or poor structuring:* When the specified features do not facilitate the reading, understanding and do not clearly states their relationship – e.g. one feature does not specify the name of the respective sub-domain or the feature specification document does not organized by sub-domain.
- *Unnecessary information or over specification:* When the specification provides more details than it is required – e.g. when it brings information regarding the later phases of the development cycle of the product line (and anticipating decisions).
- *Business rule:* Situation where the definition of the domain business rules is incorrectly specified.

Table 2 shows three feature specification examples in order to illustrate some of the feature non-conformities. Firstly *Feature non-conformity example 1* shows a specification without identification (*ID*). It is an example of *incompleteness* non-conformity, since the feature identifier was omitted. It also shows *non-traceability*, since it is difficult to establish traceability without a unique identifier. The absence of the Sub-domain name in the feature specification is another example of *non-traceability*. According to the SPL project studied, it is mandatory to specify the name of the sub-domain for all features in order to keep track of the features and their respective sub-domains. Thus, the absence of the sub-domain name falls into the *incompleteness, non-traceability*, as well as *non-Organization* non-conformity categories. Consider now the scenario in which feature *Agenda* is classified as optional, and with an exclusionary relationship with feature *Scheduling*. Consider also that feature *Scheduling* is Mandatory and requires feature *Agenda*. This

Table 2
Examples of feature non-conformities.

(a) Feature non-conformity example 1	
ID:	Name: Agenda
Description:	Manage all agendas of the system.
Priority:	Medium
Binding time:	Compile time
Feature type:	Optional
Parent feature:	none
Child feature:	none
Required feature:	none
Exclusionary feature:	Scheduling
Sub-domain:	none
(b) Feature non-conformity example 2	
ID: F002	Name: Scheduling
Description:	Manage the agenda booking of the system using google calendar application.
Priority:	High
Binding time:	Compile time
Feature type:	Mandatory
Parent feature:	Agenda
Child feature:	none
Required feature:	Agenda
Exclusionary feature:	none
Sub-domain:	none
(c) Feature non-conformity example 3	
ID: F003	Name: OS
Description:	Manage the OS of the system.
Priority:	Medium
Binding time:	Run time
Feature type:	Optional
Parent feature:	none
Child feature:	none
Required feature:	Agenda
Exclusionary feature:	none
Sub-domain:	none

scenario shows *inconsistency* non-conformity, since a mandatory feature cannot require an optional feature, neither a feature can require a second feature (Scheduling requires Agenda) where the second feature specifies an exclusionary relationship with the first feature (Kang et al., 1990; Maßen et al., 2004).

In the *Description* field of feature *Scheduling* the information "In the stretch of *Description* of feature Scheduling" comprehends the *unnecessary information* non-conformity, since information about feature implementation and coding is not required to feature specification.

In fields *Name* and *Description* of feature *OS* the *ambiguity* non-conformity is present, since the *OS* term can be interpreted in many different ways, such as Operational System, Order of Service (correct term) and so on.

5.1.3. RQ1.3. How are the non-conformities distributed in relation to the items of the feature specification template?

The idea behind this question is to investigate which items of the feature specification template have the largest number of non-conformities. It can help in the identification of which items in the template require more attention in future specifications. It can also help in the investigation of possible reasons for the occurrence of the non-conformities.

5.2. RQ2. Is there any correlation between the feature information and the non-conformities?

This question aims at investigating whether there is significant correlation between some feature information and the occurrence of non-conformities. Some predefined perspectives were considered:

- **Numbers of features and non-conformities:** This perspective investigates the effect (impact) caused by the addition of a new feature on the total amount of non-conformities.
- **Feature types and non-conformities:** It investigates if there is correlation between feature types and the amount of non-conformities.
- **Feature hierarchy and non-conformities:** It investigates if there is correlation between hierarchical features and non-conformities.
- **Feature interaction and non-conformities:** This perspective investigates the possible correlation between feature interaction and non-conformities.

Based on these perspectives, this question is composed of four subquestions, as described next.

5.2.1. RQ2.1. Are new non-conformities inserted into the project when new features are added?

This subquestion is focused on analyzing whether there is correlation between the number of features and the amount of non-conformities (studied variables). Besides, it verifies if the addition of new features into a project can lead to the inclusion of new non-conformities. The *Spearman rank correlation coefficient* measure (Gautheir, 2001) and the *Poisson regression* technique (Khoshgoftaar et al., 2001) will support the analyses performed in order to investigate this question.

5.2.2. RQ2.2. Is there any correlation between feature types and the amount of non-conformities?

This sub-question analyzes the results from the *feature types vs. non-conformities* perspective. It describes in details the distribution of non-conformities according to the different feature types, and investigates if there is a significant correlation between them.

5.2.3. RQ2.3. Is there any correlation between feature hierarchy profiles and non-conformities?

During the feature specification activity, the features can be organized according to a hierarchy, as described in (Kang et al., 1990), which establishes parental-relationships among features. In this sample, some features play the role of both parent and child simultaneously. In this study they were classified as parent-child features, for they follow both profiles.

The main motivation for this question is to investigate if there is any correlation between the features that follow some hierarchical profile and the occurrence of non-conformities. This corresponds to the *feature hierarchy and non-conformities* perspective.

5.2.4. RQ2.4. Is there any correlation between feature interactions and non-conformities?

In the feature specification activity, the features were specified, when necessary, with some interactions. These scenarios include two situations: (1) one feature requires the existence of another feature (because they are interdependent), and (2) one feature is mutually exclusive with another (they cannot coexist) (Kang et al., 1990; Cavalcanti et al., 2011b).

This research question investigates if there is significant correlation between features having interactions and the occurrence of non-conformities (*feature interaction and non-conformities* perspective).

5.3. RQ3. Is there any correlation between feature sub-domain information and non-conformities?

This question aims at investigating if there is correlation between the data regarding the sub-domain of a feature and the occurrence of non-conformities. This research question is composed of two subquestions, as follows.

5.3.1. RQ3.1 Are the sub-domains with a high number of features responsible for a high number of non-conformities?

It aims at investigating if any correlation can be established in terms of the number of features and the number of non-conformities by sub-domains, i.e. the more features a sub-domain contains, the more non-conformities will be identified.

5.3.2. RQ3.2 Is there any correlation between the sub-domain information available during the scoping phase and the occurrence of non-conformities?

In the scoping phase of the SPLSmart project, a workshop was held with the domain experts from the organization. In this workshop, the sub-domains were qualified by these experts, in terms of some attributes, such as: *experience, risk, volatility, maturity, existing code, market potential, potential reuse*, and *coupling and cohesion* (Balbino et al., 2011). The Spearman rank correlation test (Gautheir, 2001) was applied in order to investigate if there is significant correlation between qualifications of sub-domains and the non-conformities. Based on the results of the correlation test, attributes *experience*, *risk* and *volatility* were selected to investigate the estimated values of non-conformities to these attributes of sub-domains.

Experience: Indicates the level of experience that the workshop participants have regarding the sub-domain. The level of *experience* can be:

- **High:** participants have a complete understanding and some stakeholders participated in projects related to the sub-domain.
- **Partial:** the sub-domain is known, but the understanding is partial.
- **Low:** the participants do not know the sub-domain.

Risk: Risks are identified and analyzed to determine their negative impact on the sub-domain. In the analysis, the risks are prioritized according to the staff perception about their severity. The *risk impact* can be:

- **High:** potential problems will occur and will be difficult to manage.
- **Relevant:** any problems can occur, however, they can be managed.
- **Low:** no apparent problems.

Volatility: Determines whether the sub-domain can change over time. The *Volatility* can be:

- **High:** constant changes happen in the sub-domain.
- **Average:** possible changes can be expected, but there is no major impact.
- **Low:** few changes can occur.

5.4. RQ4. How much effort is spent during the inspection of feature specifications? How is it distributed in terms of inspection tasks?

All participants were asked to report the individual time spent in every inspection task, as listed in the previous section. This data represents the effort in man-hours required to perform the tasks.

This analysis can provide a quantitative reference about the dimension and execution complexity of every inspection task, serving as a good indicator of which task need to be improved in the process. It can also provide some insights about baseline values to serve as estimative for required effort in next iterations.

6. The inspection approach

This work's main focus is not to present an inspection process for SPL phases and artifacts, however a brief overview is provided in this section.

The approach is based on Fagan's inspection process (Fagan, 1976) and inspection practices (Yourdon, 1989; Gilb and Graham, 1993; Wheeler et al., 1996; Wiegers, 2002; Parnas, 2004; Thelin et al., 2004; Kollanus and Koskinen, 2009). Six software engineers use this approach: three reviewers, one moderator, and two readers/document authors. Each one was responsible for, at least, one role, as detailed next:

- *Moderator:* leads and manages the tasks related to inspection activities.
- *Reader:* reads the artifact during the inspection meeting.
- *Reviewer:* verifies the artifacts under inspection and identifies the defects or non-conformities.
- *Author:* responsible for creating the artifact under inspection.

In this study, the author of the artifact under inspection worked also as *reader* in the inspection meeting. In order to systematically inspect the feature specifications, a set of tasks were defined to guide the inspection activity, such as:

- *Planning:* in this task, the roles are selected and assigned to form the inspection team. Support materials and documents are prepared, selected and sent to the inspection team. Moreover, inspection activities are scheduled and booked.
- *Preparation:* the moderator checks the artifact under inspection and sends it to the reviewers. The reviewers individually inspect the artifacts in order to capture non-conformities.
- *Meeting:* the non-conformities captured in the preparation activity are discussed in order to identify and discard false non-conformities. The output of this task is the consolidated list of non-conformities.
- *Correction:* the authors of the artifact under inspection receive the list of non-conformities and perform the correction activity.
- *Validation:* the moderator validates the artifacts in order to verify if the author properly fixed the non-conformities after the inspection meeting.

In order to support the proposed inspection approach, we defined the following inspection instruments:

- *Quality Attributes List:* Set of quality attributes required for feature specifications, such as completeness, correctness and consistency.
- *Non-Conformities List:* Set of non-conformities based on the quality attributes, such as incorrectness, incompleteness and inconsistency.
- *Checklist:* Guidance instrument to support the inspections, helping the reviewers to identify non-conformities.
- *Inspection Plan:* This is responsible for orchestrating the inspections, defining the objective, the stakeholder's roles, instruments, artifacts to be inspected, and so on.
- *Non-Conformity Records:* Spreadsheet for recording all the non-conformities found during the inspections activities.
- *Inspection Report:* The document in which the non-conformities found are reported, and recommendations can be made to improve the quality of the artifact under inspection.

Table 3
Inspection activity executions.

Artifact	Iteration	Period	Data
Features specification	1st	August	61 features 88 h
	2nd	November	31 features 51 h

Table 4
Features non-conformities per sub-domain.

Sub-domains	Num. features	Num. non-conformities	Non-conformity densities
Patient	4	4	1.00
Reception	22	33	1.50
Doctor's office	23	31	1.35
Billing	8	20	2.50
Scheduling	4	8	2.00
Supply management	11	13	1.18
Inventory	3	3	1.00
Purchase management	8	7	0.88
Laboratory	9	18	2.00

Fig. 3 summarizes the main elements of the inspection approach applied in this study. More in detail, as soon as the artifact is built, the *author* sends the feature specification to the *moderator*, together with other complementary work products, when necessary. The *moderator* checks the artifacts to be inspected and sends them to the *reviewers*. All *reviewers* individually review the artifact guided by the *checklist*, classify the non-conformities found according to the *non-conformities list* and record them in the *non-conformities records* spreadsheet. The spreadsheet is forwarded to the *moderator* by *reviewers* with the recorded information, non-conformities, the time spent during the individual review and some notes. After gathering all review information, a meeting is scheduled in order to solve some disagreements or suggest ways to fix the non-conformities. In this meeting, the *reader* reads the documents (features specifications and complementary artifacts), the *reviewers* ask the authors about doubts and all meeting participants can discuss the existing problems in the artifacts under inspection. After the *moderator* gathers all information about the non-conformities, he sends to the *author* the *inspection report*. Then, the *author* fixes the non-conformities and sends the new version of the work product to the *moderator*. Finally, the *moderator* checks the performed changes, to be confident that they were properly corrected.

7. Results of the empirical study

In this section, we report the findings of the empirical study. Table 3 shows some quantitative data. Some exploratory analyses (Wainer, 2007) were performed to gain more insight and also come up with ideas and hypotheses for new research. Based on the gathered data, the analysis was performed to answer the research questions and try to understand possible directions for further research.

The study analyzed a representative sample of the specified features for the SPLSmart project in the scoping phase. In this sample, 92 specified features were inspected. The inspection of these feature specifications took around 139 man-hours, and the number of features per sub-domain can be seen in Table 4.

The analyzed dataset comprises data from the first two iterations with 9 sub-domains of the SPLSmart project. The first iteration is composed of the sub-domains:

- Patient: It handles the routines regarding patient registration data.

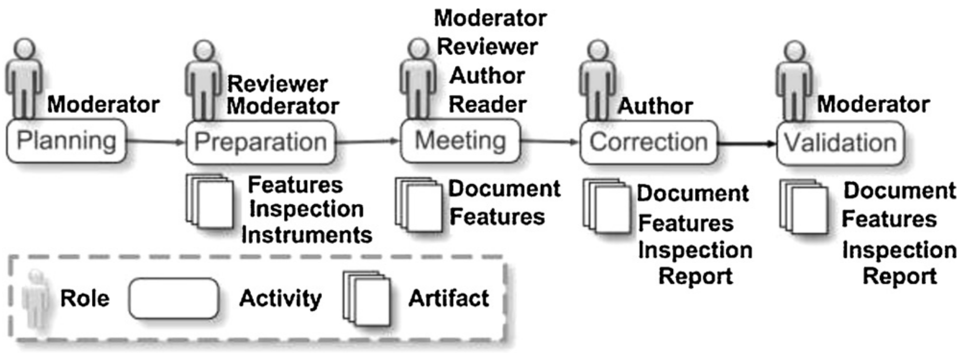


Fig. 3. Systematic inspection approach on feature document.

- Reception: It involves the tasks to carry out the routines of the medical unit's front office, such as check-in of patients, patient attendance order, and clinical procedures scheduling (appointment, history and physical examination, and surgery).
- Doctor's office: It handles patients' medical histories, including updating and maintaining medical records.
- Billing: It deals with the processing and generation of accounts receivable and accounts payable.
- Scheduling: It is responsible for scheduling patient's doctor appointments, like examinations, surgeries, etc.

The second iteration is composed by:

- Supply management: It manages medical supply data and records.
- Inventory: It is responsible for managing medical supplies inventory, controlling the inputs, and outputs, and balance for requesting new materials.
- Purchase management: It is responsible for managing the process and routines for purchasing medical supplies.
- Laboratory: It is responsible for managing the operational routines of medical laboratory units, such as laboratory services, patient laboratory admission and attendance order, laboratory tests results, etc.

For this study, we consider a **Feature** as being "a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems" (Kang et al., 1990). The features define both common aspects of the domain, as well as differences among related products in the domain. It is also used to define the domain in terms of mandatory, optional or alternative characteristics of the related products.

Following, the research questions of the study are discussed.

7.1. RQ1. What is the distribution of non-conformities in the project?

This question was divided into three subquestions. Their answers are presented next.

7.1.1. RQ1.1. How are the non-conformities spread in terms of sub-domains?

The analysis considers the relationship among the number of features (NFeatures) and the number of non-conformity occurrences found (NNonConformities), establishing the value of non-conformity density or Density (Eq. (1)) per sub-domain. This value was analyzed using the Control Chart (Montgomery and Runger, 2006).

$$Density = \frac{NNonConformities}{NFeatures} \quad (1)$$

The control chart is a statistical process control very useful when applied to software engineering quality control. It is an instrument that provides information about process capability and diagnostic (Montgomery and Runger, 2006; Zhang and Kim, 2010). We used the control chart to plot the non-conformities density per sub-domain to observe the behavior between the non-conformity density variations in the confidence interval provided by the control chart limits (UCL and LCL). The confidence interval is the interval between the UCL and LCL limits. Thus if C_i denotes the non-conformity density obtained in the ith observation, the control chart plots the data points at the height $C_1, C_2, \ldots, C_n$. The control chart also has a Center Line (CL) at height μC (the average of C_i) and the following 3σ control lines, where UCL (Eq. (2)) is the Upper Control Limit and LCL (Eq. (3)) is the Lower Control Limit. It is important to highlight that, if LCL is a negative value, it is set to 0 (Montgomery and Runger, 2006; Zhang and Kim, 2010) as happened with the distribution shown in Fig. 4.

$$UCL = \mu C + 3\sqrt{\mu C} \quad (2)$$

$$LCL = \mu C - 3\sqrt{\mu C} \quad (3)$$

Observing the control chart of Fig. 4 (data from Table 4), it is possible to see the distribution of the non-conformity densities per sub-domains. The control chart has the following approximate values for attributes: CL = 1.49, UCL = 5.15 and LCL = −2.17 (set to 0). From the nine points (sub-domain non-conformity densities) plotted, four points were plotted above the center line and five points plotted below it. It is possible to observe a random behavior pattern on the control chart. Thus, the control chart shows that the non-conformity densities for the sample sub-domains are in control for the confidence interval computed for this distribution (Montgomery and Runger, 2006).

It is important to highlight that although the sub-domain *Billing* did not present the highest number of features nor the highest number of non-conformities in the study under investigation, it had the

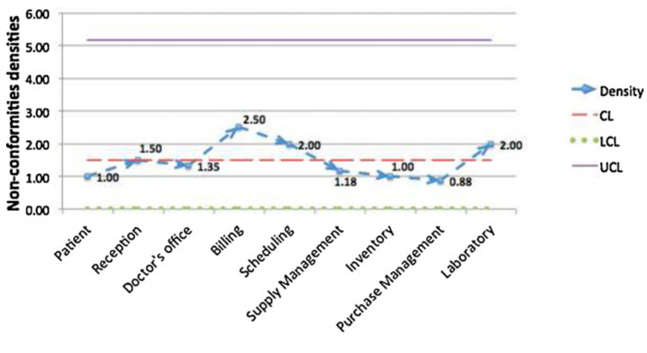


Fig. 4. Control chart – feature non-conformity densities per sub-domain.

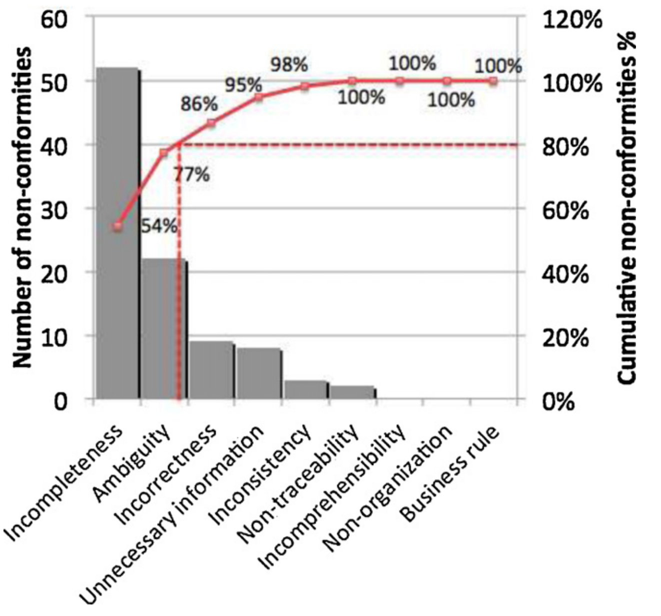


Fig. 5. Pareto's chart for the distribution of feature non-conformity types – 1st iteration.

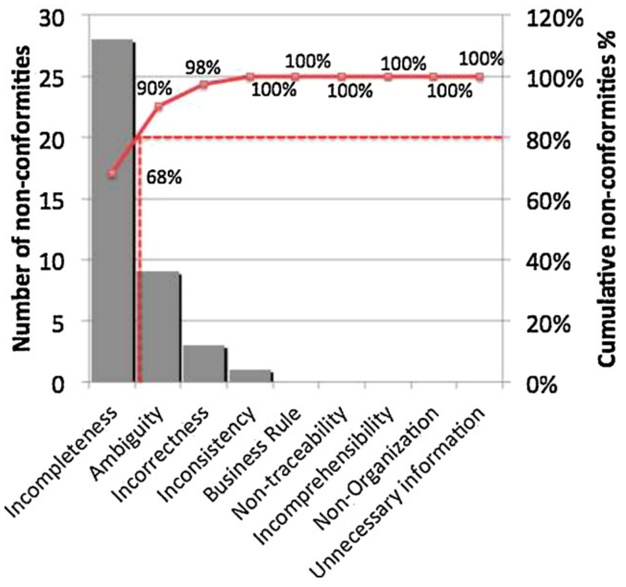


Fig. 6. Pareto's chart for the distribution of feature non-conformity types – 2nd iteration.

highest feature non-conformities density (Table 4). On the other hand, whereas the company classified sub-domain *Billing* with high experience and low volatility, it also classified *Billing* as having high risk (Table 13). In this direction, the analysis of the RQ3.2 shows that for those sub-domains classified with high risk, a higher number of feature non-conformities was estimated, using Poisson nonlinear regression prediction models. All these results are the first step toward establishing a baseline of non-conformity densities in the specification of features in the SPL project.

7.1.2. RQ1.2. What is the distribution of non-conformity occurrences on types of non-conformities?

We classified the reported amount of non-conformity occurrences and applied Pareto's principle, which can statistically identify the main causes of occurrences (Gallegati, 2008). Pareto's principle (Gallegati, 2008), also known as 80–20 rule, states that roughly 80% of the effects come from 20% of the causes. The Pareto graphs (Figs. 5 and 6) show the individual values in descending order (bars). The line represents the cumulative total.

By analyzing the Pareto graphs (Figs. 5 and 6), *incompleteness*, *ambiguity* and *incorrectness* represent the most identified non-conformity types for the two iterations of the analyzed features.

Comparing the results from the two iterations, we can observe that the non-conformity types *Unnecessary information* and *Non-traceability* do not have occurrences in the second iteration. The percentage of *Ambiguity, Incorrectness* and *Inconsistency* non-conformity occurrences decreased in the second iteration. On the other hand, the percentage of *Incompleteness* non-conformities increased (54 to 68%), and therefore it deserves more attention in next iterations.

Analyzing the percentage reduction related to the numbers of non-conformities for the types of non-conformity – *Ambiguity, Incorrectness* and *Inconsistency* – we believe that the reason for this reduction was the learning effect (Pezze and Young, 2007) on the authors of the feature specifications, between the first and second iterations. In the end of each sub-domain inspection activity, the authors received the inspection report document, which contained the non-conformities detected on the artifact under inspection and recommendations for their improvement.

On the other hand, the percentage of *Incompleteness* non-conformities increased for feature specification between the first and second iterations. This can be related to the learning effect of the reviewers, because they can learn more about the project domain in each inspection activity round. Nevertheless, both issues need more studies.

7.1.3. RQ1.3. How are the non-conformities distributed in relation to the items of the feature specification template?

An exploratory analysis was performed in order to identify which feature template items have the highest occurrence of non-conformities and the possible reasons. Table 5 shows the non-conformity occurrences distribution for the template items per sub-domain. It is possible to observe that *name*, *description* and *required feature* template items are highlighted, because they registered, respectively, 24, 65 and 38 non-conformity occurrences, for all studied sub-domains.

Thus, the first analysis task was to identify which types of non-conformities were responsible for such distribution. In this way, an exploratory analysis was performed and it was possible to understand the reasons for such values in the template items (Table 6).

For the *name* item, the non-conformity types with more occurrences were Incompleteness and Ambiguity. On the other hand, for the *description* item, Incompleteness and Ambiguity were also the types with most occurrences. Finally, for the *Required feature* item,

Table 5
Feature non-conformity distribution: template items by sub-domains.

	Sub-domains								
	A	B	C	D	E	F	G	H	I
Name	1	8	6	5	2	0	0	0	2
Description	3	20	8	6	3	9	3	3	10
Priority	0	0	0	1	0	0	0	0	0
Binding time	0	1	1	0	0	1	0	0	0
Required feature	0	1	16	8	3	0	0	4	6
Excluded feature	0	0	0	0	0	0	0	0	0
Parent feature	0	2	0	0	0	0	0	0	0
Child feature	0	1	0	0	0	3	0	0	0

Sub-domains: A – Patient, B – Reception, C – Doctor's office, D – Billing, E – Scheduling, F – Supply management, G – Inventory, H – Purchase management, I – Laboratory.

Table 6
Feature non-conformity distribution: template items by non-conformity types.

	First and second iteration							
	Name	Description	Priority	Binding time	Required feature	Excluded feature	Parent feature	Child feature
Incompleteness	12	32	0	0	32	0	2	2
Ambiguity	9	22	0	0	0	0	0	0
Incorrectness	3	3	1	3	1	0	0	1
Unnecessary information	0	8	0	0	0	0	0	0
Inconsistency	0	0	0	0	3	0	0	1
Non-traceability	0	0	0	0	2	0	0	0

incompleteness was the non-conformity type with most occurrences.

As it can be seen, *Incompleteness* was the type with most occurrences in items *name, description* and *required feature*. This first finding reinforces the finding stated in the results of question RQ1.2 (where the question highlighted the incompleteness non-conformity type). Hence, this effect can serve as an input for the development of a quality-improving plan in the feature specifications for the next iterations and projects.

Observing the distribution of the features that recorded non-conformities (defective features) in the template items (Fig. 7), it is possible to see that the *Description* template item had the largest number of defective features. Hence, 56% of the features reported non-conformity occurrences in the *Description* item. This reinforces the finding that *Description* was the most problematic item of template, as stated beforehand in this work.

7.2. RQ2. Is there any correlation between the feature information and the non-conformities?

On this question investigation, the significant correlations were tested using the Spearman rank correlation coefficient with significance level of 5% (p-value). Spearman correlation coefficient or Spearman ρ is a nonparametric measure for evaluating the degree of correlation between two variables (Gautheir, 2001; Zhang et al., 2010). In this question, the variables represent the items investigated in each of the four investigated perspectives. For example, for perspective *Numbers of features and non-conformities*, one variable is the number of features (X) and the other is the number of feature non-conformities (Y). The result of Spearman correlation test is represented by the Greek letter ρ (rho). ρ can have a value between -1 and 1. A positive ρ value means that there is a positive correlation, i.e. the number of non-conformities (Y) tends to increase when the number of features (X) increases. A $\rho = 1$ means that there is a perfect positive correlation. A negative ρ value means that there is a negative correlation, i.e. the number of non-conformities (Y) tends to decrease when the number of features (X) increases. A $\rho = -1$ means that there is a perfect negative correlation. On the other hand, a ρ value = 0 or close to 0 means that there is no tendency for Y to either increase or decrease when X increases. So, there is no significant correlation between the analyzed variables.

The Poisson regression analysis (Khoshgoftaar et al., 2001) is a form of analysis used to model count data and contingency tables. It has a response variable that has a Poisson distribution, and assumes that the logarithm of its expected value can be modeled by a linear combination of unknown parameters. In this question, the response variable ($\hat{Y}$) is the intercept estimated value calculated from the distribution of feature non-conformities and the unknown parameters are estimated values calculated from distributions of the number of features, features with hierarchical role and features that have interaction with other features. The linear combination is composed of intercept value ($\hat{\beta}_0$) combined individually with the exponential function (e) of the unknown parameter ($\hat{\beta}_1$) and estimated amount of features, independent variable (X), as can be seen in Eq. (4). For the multiple Poisson regression analysis, the linear combination is composed of the intercept value and the exponential function of the unknown parameters under analysis, as can be seen in Eq. (5).

$$\hat{Y} = \hat{\beta}_0 \cdot e^{\hat{\beta}_1 \cdot X} \tag{4}$$

$$\hat{Y} = \hat{\beta}_0 \cdot e^{(\hat{\beta}_1 \cdot X_1)} \cdot e^{(\hat{\beta}_2 \cdot X_2)} \cdot \ldots \cdot e^{(\hat{\beta}_n \cdot X_n)} \tag{5}$$

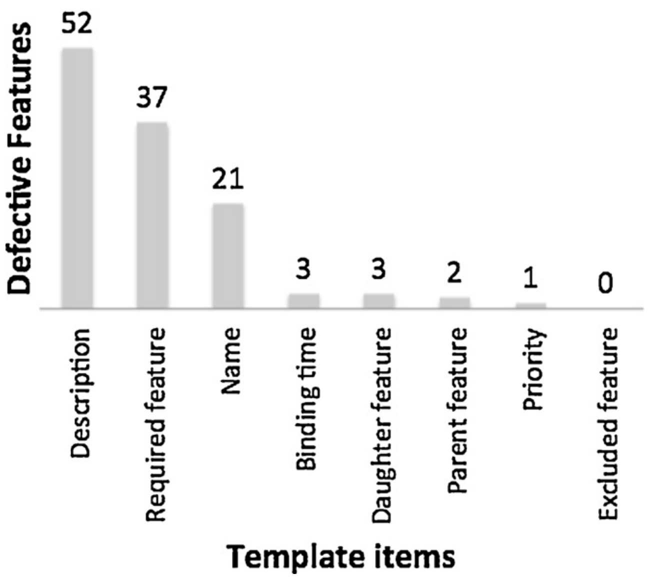


Fig. 7. Distribution of defective features in the template items.

Tables 7–9 show, respectively, the coefficient and variables values of Spearman rank tests; the parameter values of the simple Poisson regression analysis; and the candidate and selected parameters values of the Poisson regression multiple model. These data were used for performing the analysis that supports the answers to the four subquestions, as discussed next.

Table 7
Spearman rank correlation parameters: unit of analysis – inspection on features specifications.

Correlation	ρ	S-statistic	p-Value
Total features and num. non-conformities	0.8439	18.7333	0.0042
Total required features and num. non-conformities	0.8254	20.9573	0.0062
Total requesting features and num. non-conformities	0.7419	30.9670	0.0221
Total parent features and num. non-conformities	0.3889	73.3293	0.3009
Total child features and num. non-conformities	0.3593	76.8824	0.3423

Table 8
Simple Poisson regression parameters: unit of analysis – inspection on features specifications.

Parameters	Estimate	Std. error	*z*-Value	*p*-Value	
Intercept	1.72	0.18	9.31	<2e−16	***
Num. features	0.08	0.01	7.27	3.59e−13	***
AIC: 62.351					
Intercept	2.20	0.14	16.09	<2e−16	***
Num. required features	0.13	0.02	5.69	1.27e−08	***
AIC: 86.837					
Intercept	1.23	0.30	4.13	3.65e−05	***
Num. requesting features	0.28	0.05	5.70	1.20e−08	***
AIC: 77.535					
Intercept	2.29	0.13	17.17	<2e−16	***
Num. parent features	0.50	0.10	5.03	4.84e−07	***
AIC: 89.22					
Intercept	2.52	0.10	24.46	<2e−16	***
Num. child features	0.06	0.01	4.52	6.08e−06	***
AIC: 96.803					
Intercept	2.02	0.18	11.04	<2e−16	***
High risk sub-domains	1.26	0.23	5.53	3.3e−08	***
Relevant risk sub-domains	1.16	0.23	5.00	5.8e−07	***
AIC: 59.557					

Akaike information criterion (AIC) is a measure of the relative goodness of fit of a statistical model. Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.

7.2.1. RQ2.1. Are new non-conformities inserted into the project when new features are added?

The results for *the number of features and the number of non-conformities* indicated that there is a significant positive correlation, since the Spearman correlation coefficient resulted in $\rho \approx 0.84$ (Table 7). The Poisson regression investigation about the number of features and number of non-conformities (Table 8) presents the following estimated values: $\hat{\beta}_0 = 1.72$, $\hat{\beta}_1 \approx 0.08$. Based on Eq. (4) and calculating these values for a new feature (Eq. (6)), the result value of $\hat{Y} = 1.86$ is found. This $\hat{Y}$ value represents, based on the analyzed sample and in the Poisson regression model assembled, that the addition of a new feature results in the addition of 1.86 new non-conformities occurrences.

$$\hat{Y} = 1.72 \cdot e^{0.08 \cdot 1} \tag{6}$$

Table 9
Multiple Poisson regression models: unit of analysis – inspection on features specifications.

Parameters	Estimate	Std. error	*z*-Value	*p*-Value	
(a) Multiple Poisson regression – candidate model					
Intercept	1.5724	0.2512	6.259	3.88e−10	***
Num. features	0.0744	0.0351	2.118	0.0342	*
Num. required features	0.0082	0.0586	0.139	0.8893	
Num. parent features	−0.2535	0.2092	−1.212	0.2256	
High risk sub-domains	0.7405	0.3059	2.421	0.0155	*
Relevant risk sub-domains	0.6116	0.3435	1.781	0.0750	.
AIC: 53.637					
(b) Multiple Poisson regression – selected model					
Intercept	1.7085	0.2085	8.194	2.53e−16	***
Num. features	0.0455	0.0144	3.150	0.0016	**
High risk sub-domains	0.8364	0.2717	3.078	0.0021	**
Relevant risk sub-domains	0.6919	0.2833	2.442	0.0146	*
AIC: 51.166					

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.

7.2.2. RQ2.2. Is there any correlation between feature types and the amount of non-conformities?

The studied feature sample is composed of 6 optional features and 86 mandatory features. Analyzing the relationship between the feature types and the non-conformities (Table 10), we identified that for only one optional feature there was no non-conformity occurrences. On the other hand, the remaining optional features recorded 10 non-conformities. The non-conformity density (Eq. (1)) for optional features ≈ 1.66, and the non-conformities identified for optional features were classified as following: *5 ambiguity and 5 incompleteness occurrences*. Among the optional features that registered non-conformities, 4 features registered incompleteness and ambiguity in the *Description* template item, 3 features had incompleteness in the *Required feature* item and 1 feature recorded ambiguity in the *Name* item.

Based on the descriptive analysis above, it was possible to identify that template items *Description* and *Required features* are the items with the highest number of non-conformities for optional features. In addition, the main non-conformity types captured for optional features are *Ambiguity* and *Incompleteness*.

Regarding **mandatory features**, 19 features did not present non-conformities and **67 mandatory features recorded 127 non-conformities occurrences**. This represents a 1.48 non-conformity density for mandatory features. As in the case of optional features, we analyzed the non-conformities related to mandatory features. It was distributed as following (Table 10): 75 incompleteness occurrences, 26 ambiguity occurrences, 12 incorrectness occurrences, 8 unnecessary information occurrences, 4 inconsistency occurrences and 2 non-traceability occurrences. Table 11 shows the distribution of feature non-conformity in the template of mandatory features. The table highlights incompleteness occurrences in most items of feature template.

For the mandatory features with non-conformity occurrences, 48 features recorded incorrectness, incompleteness, ambiguity and unnecessary information for the *Description* template item; 34 features had incorrectness, incompleteness, inconsistency and

Table 10
Features non-conformities distribution: features types per type of non-conformity.

	Non-conformity types					
	Incompleteness	Ambiguity	Incorrectness	Inconsistency	Non-traceability	Unnecessary information
Optional features	5	5	0	0	0	0
Mandatory features	75	26	12	4	2	8

Table 11
Non-conformities distribution per feature template item for mandatory features.

	First and second iteration							
	Name	Description	Priority	Binding time	Required feature	Excluded feature	Parent feature	Child feature
Incompleteness	12	30	0	0	29	0	2	2
Ambiguity	8	18	0	0	0	0	0	0
Incorrectness	3	3	1	3	1	0	0	1
Unnecessary information	0	8	0	0	0	0	0	0
Inconsistency	0	0	0	0	3	0	0	1
Non-traceability	0	0	0	0	2	0	0	0

non-traceability for the *Required feature* item; 20 features recorded incorrectness, incompleteness and ambiguity for the *Name* item; 3 features recorded incorrectness, incompleteness and inconsistency for the *Child feature* item; 3 features had incorrectness for the *Binding time* item; 2 features recorded incompleteness for the *Parent feature* item and 1 feature had incorrectness for the *Priority* item.

Table 12 presents the distribution of non-conformity occurrences in the project iterations. Analyzing the non-conformity data from the mandatory features, we observed that 96% of mandatory features that recorded non-conformities in the *Name* template item were specified and inspected during the first iteration. Considering the amount of mandatory features specified in the first iteration (59 features), this number is significant, because it means that for 32% of the mandatory features specified in the first iteration, non-conformities were identified for the *name* template item. Considering only the mandatory features that registered non-conformity occurrences in the first iteration (45 features), this ratio is higher than 42%, indicating that the *Name* item was the main responsible for most of the non-conformities generated for mandatory features in the first iteration. On the other hand, we noticed that only 1 mandatory feature recorded non-conformity occurrence for *Name* item in the second iteration. This can represent a significant improvement in the feature specification activity between the first and second iterations. It can be also related to the establishment of standards for feature naming after the first iterations.

Comparing the non-conformity densities from optional and mandatory features, we can observe that optional (1.66) features had a higher non-conformity density than mandatory features (1.48). Thus, the notion of variability in the context of the SPL scoping phase, represented by optional features, generated more non-conformities per number of features, on the analyzed sample.

7.2.3. *RQ2.3. Is there any correlation between feature hierarchy profiles and non-conformities?*

In the sample, 26 from the 92 specified features could be classified as having a hierarchical feature role (Kang et al., 1990). From this subset, 5 features were classified as *Parent feature*, 19 as *Child feature* and 2 as *Parent-Child feature*.

All the 5 **Parent features** had non-conformities, with 10 occurrences each. The non-conformity density for parent features was 2.0 and they were classified and distributed as: 6 incompleteness occurrences, 2 ambiguity occurrences, 1 inconsistency occurrence, and 1 non-traceability occurrence. By observing the distribution of non-conformities in the feature specification template items, it was identified: 1 feature had incompleteness for the *Name* template item, 4 features had incompleteness and ambiguity for the *Description* item, 2 features had inconsistency and non-traceability for the *Required feature* item, and 1 feature had incompleteness for the *Child features* item.

Regarding the 19 **Child features** specified, 5 features did not record any non-conformity, and 14 features recorded 25 non-conformity occurrences. Their non-conformity density was 1.79. The non-conformities were classified as follows: 19 incompleteness occurrences, 3 ambiguity occurrences, 2 unnecessary information occurrences, and 1 incorrectness occurrence. By observing the distribution of non-conformities in child features specification template items: 10 features recorded incompleteness for the *Required feature* item, 7 features recorded incompleteness, ambiguity and unnecessary information for the *Description* item, 5 features had incompleteness, ambiguity and incorrectness for the *Name* item, and 2 features recorded incompleteness for the *Parent feature* item.

For the 2 **parent-child features** specified, 1 feature did not record any non-conformity and 1 feature had 1 incompleteness non-conformity in the *Required feature* template item.

Analyzing all the 26 features with any hierarchical profile (parent, child or parent-child feature), 20 features did not record any non-conformity. The *required feature* template item showed the largest number of defective features (13 features), whereas *description* had more non-conformities (14 occurrences). These findings partially reinforce the results of question RQ1.3. Regarding to the distribution of non-conformity types (Fig. 8), 26 out of the 36

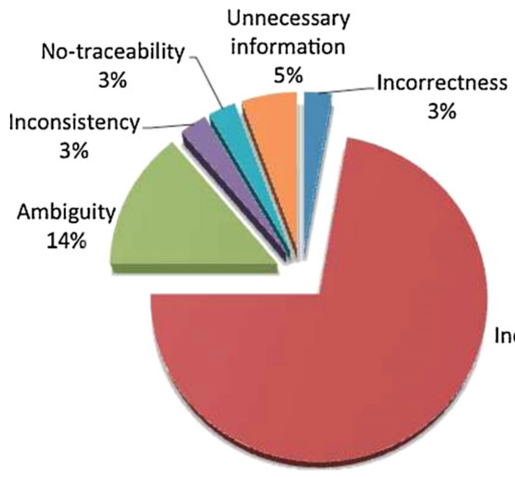


Fig. 8. Distribution of hierarchical feature non-conformities in the type of non-conformities (%).

Table 12
Mandatory Feature non-conformities distribution: items of template per type of non-conformity.

	Incompleteness	Ambiguity	Incorrectness	Inconsistency	Non-traceability	Unnecessary information
First iteration						
Name	11	8	3	0	0	0
Description	15	13	3	0	0	8
Priority	0	0	1	0	0	0
Binding time	0	0	2	0	0	0
Required feature	23	0	1	3	2	0
Excluded feature	0	0	0	0	0	0
Parent feature	2	0	0	0	0	0
Child feature	1	0	0	0	0	0
Second iteration						
Name	1	0	0	0	0	0
Description	15	5	0	0	0	0
Priority	0	0	0	0	0	0
Binding time	0	0	1	0	0	0
Required feature	6	0	1	0	0	0
Excluded feature	0	0	0	0	0	0
Parent feature	2	0	0	0	0	0
Child feature	1	0	1	1	0	0

recorded non-conformity occurrences were classified as *incompleteness*. This finding partially reinforces the results of question RQ1.2.

In addition, in the analysis related to feature hierarchy profiles and non-conformities we observed that **for parent and child features** there is no significant correlation, as the obtained Spearman coefficient values were, respectively, $\rho = 0.39$ and $\rho = 0.36$ (Table 7). Nevertheless, the Poisson regression analysis presented the estimated values: $\hat{\beta}_0 \approx 2.29$ (parent feature) and 2.52 (child feature), $\hat{\beta}_1 \approx 0.50$ (parent feature) and 0.06 (child feature). Based on Eq. (4) and calculating these values for one new parent and child feature (Eqs. (7) and (8)), we obtained the result values of $\hat{Y} \approx 3.76$ (parent feature) and 2.67 (child feature). This $\hat{Y}$ value represents, based on the analyzed sample for Poisson regression model assembled, that the addition of a new feature with hierarchical role would cause the addition of 3.76 non-conformities for *parent* features and 2.67 non-conformities for *child* features.

$$\hat{Y} = 2.29 \cdot e^{0.50 \cdot 1} \tag{7}$$

$$\hat{Y} = 2.52 \cdot e^{0.06 \cdot 1} \tag{8}$$

7.2.4. RQ2.4. Is there any correlation between feature interactions and non-conformities?

In the dataset, 59 features had some kind of interaction with other features (Kang et al., 1990). No feature was assigned as exclusionary feature, 30 features required other features, 17 features just are required by others and 12 features simultaneously require and are required by other features.

Considering all features mapped with this interaction, the data was collected. It reported 97 non-conformity occurrences distributed as follows: 52 *incompleteness* occurrences, 24 *ambiguity* occurrences, 10 *incorrectness* occurrences, 5 *unnecessary information* occurrences, 4 *inconsistency* occurrences, and *non-traceability* occurrences. **Incompleteness** was the non-conformity type that presented the highest volume of non-conformity occurrences for features that had some interaction.

By analyzing the distribution of non-conformities in **template items**, we observed that: 19 features recorded incompleteness, incorrectness and ambiguity in the *name* template item; 49 features reported incompleteness, incorrectness, ambiguity and unnecessary information for the *description* item; 1 feature reported incorrectness for the *priority* item; 1 feature had incorrectness for the *binding time* item; 26 features reported incompleteness, incorrectness, inconsistency and non-traceability for the *required feature* item, and 2 features had incompleteness, incorrectness and inconsistency for *child feature* item. The other feature template items did not have any non-conformity. The *description* template item had the highest number of features with non-conformities and this template item had also the highest number of non-conformity occurrences for features that had some interaction.

The correlation analysis for feature interaction profiles and non-conformity perspective indicates that there is a significant positive correlation, as the Spearman coefficient $\rho = 0.74$ for *features that require another feature and number of non-conformities* and $\rho = 0.83$ for *required features and number of non-conformities* (Table 7). The Poisson regression analysis presents the estimated values: $\hat{\beta}_0 \approx 1.23$ and $\hat{\beta}_1 \approx 0.28$ for *features that require another feature*, and $\hat{\beta}_0 \approx 2.20$ and $\hat{\beta}_1 \approx 0.13$ for *required features*. Based on Eq. (4) and calculating these values for one new feature that has some interaction (Eqs. (9) and (10)), we obtained the result values of $\hat{Y} \approx 1.62$ (features that require another feature) and 2.50 (required features). This means that based on the analyzed sample for the Poisson regression model assembled, the addition of a new feature that interacts with another would cause the addition of 1.62 non-conformities for features that require another feature and 2.50 non-conformities for required features.

$$\hat{Y} = 1.23 \cdot e^{0.28 \cdot 1} \tag{9}$$

$$\hat{Y} = 2.20 \cdot e^{0.13 \cdot 1} \tag{10}$$

In order to better understand RQ2, a complementary analysis was performed for investigating the issues of this question. We analyzed the possibility of assembling a multiple Poisson regression model for the investigated feature dataset, composed from the intercept parameter (number of non-conformities) and the candidate parameters that show significant correlation on the analysis performed in the subquestion of this question. It is possible to see in Table 9a that the combination of all the candidate parameters did not return a reliable multiple Poisson regression model, because two parameters presented significance level (*p*-value) outside the 5% range (num. required features and num. parent features). From this first result, these two parameters were removed and another multiple Poisson regression model was assembled, composed of the parameters (Table 9b): the response variable (Intercept) and the input variables (num. features, high risk sub-domains and relevant risk sub-domains). The multiple Poisson regression model for the features dataset is a prediction model that can provide the estimated amount of non-conformities from the input variables that make up the model.

Table 13
Scope phase sub-domains analysis summary.

Sub-domains	Experience	Risk	Volatility
Patient	High	Low	Low
Reception	High	High	High
Doctor's office	High	Relevant	Low
Billing	High	High	Low
Scheduling	High	Low	Low
Supply management	High	Low	Low
Inventory	High	Low	Low
Purchase management	High	Low	Low
Laboratory	High	Relevant	Low

7.3. RQ3. Is there any correlation between feature sub-domain information and non-conformities?

In order to answer this question, we split it in the following subquestions.

7.3.1. RQ3.1 Are the sub-domains with a high number of features responsible for a high number of non-conformities?

Observing the distribution of the number of features specified in the sub-domains and the amount of non-conformities (Table 4), we identified that the sub-domains that have the highest number of features are those with most non-conformities. In the first iteration, the sub-domains with highest numbers of specified features **Doctor's office** – 23 features and **Reception** – 22 features reported 31 and 33 non-conformity respectively. In the second iteration, sub-domains **Supply management** – 11 features and **Laboratory** – 9 features, respectively, reported the highest numbers of non-conformity occurrences (13 and 18).

Analyzing together all the sub-domains of the sample, we observed that the sub-domains with the highest number of specified features, **Doctor's office** and **Reception**, recorded the highest numbers of non-conformity occurrences.

7.3.2. RQ3.2 Is there any correlation between the sub-domain information available during the scoping phase and the occurrence of non-conformities?

From the selected attribute values (Table 13), we applied the Poisson regression analysis (Khoshgoftaar et al., 2001) with the goal of verifying whether there is any correlation between the selected attributes and the non-conformity occurrences.

Based on the distribution of attribute values presented in Table 13, it is possible to identify that the distribution of attributes *Experience* and *Volatility* do not have variation neither any random behavior pattern. It implies that the values from those attributes are not suitable for applying the Poisson regression. In this case, we analyzed just attribute **Risk**.

By applying the Poisson regression to the distribution from the sub-domain risk attribute, significant values were obtained, as shown in Table 8, $\hat{\beta}_0 \approx 2.01$, and $\hat{\beta}_1 \approx 1.16$ for relevant risks and 1.26 for high risks. Based on Eq. (4) and calculating these values for one new relevant or high-risk sub-domain (Eqs. (11) and (12)), we obtained the result values of $\hat{Y} \approx 6.45$ (relevant risk) and 7.12 (high risks). Thus, when one new sub-domain is qualified as **relevant risk**, it is possible to predict the addition of **6.45** non-conformity occurrences in the feature specification activity. If the new sub-domain is qualified as **high risk**, it is possible to predict the addition of **7.12** non-conformity occurrences in the feature specification activity.

$$\hat{Y} = 2.01 \cdot e^{1.16 \cdot 1} \qquad (11)$$

$$\hat{Y} = 2.01 \cdot e^{1.26 \cdot 1} \qquad (12)$$

The results indicate that the risk attribute can be a good indicator for prioritizing sub-domains in the inspection activity. They can determine which sub-domains are most critical and require more attention in order to identify non-conformities.

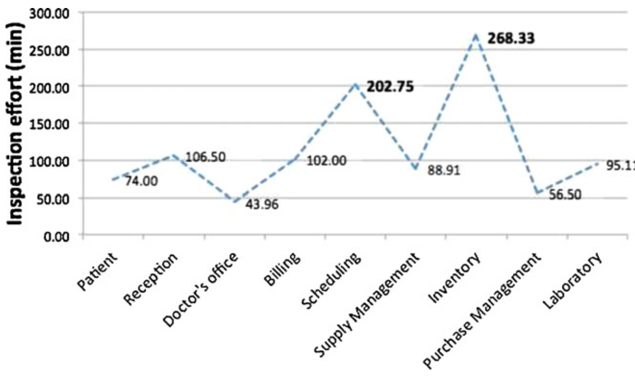


Fig. 9. Inspection effort – feature specification.

7.4. RQ4. How much effort is spent during the inspection of feature specifications? How is it distributed in terms of inspection tasks?

The data provided by the inspection activity participants, as shown in Table 14, details the time spent (minutes) in every inspection task (planning, preparation, meeting, correction, validation) in the first and second iterations. These values represent the total amount of time spent, considering the sum of individual times registered by each inspector.

As it can be seen, the **planning task** time schedule was fixed in 120 min. This represents the amount of time devoted to preparing the inspection round for every sub-domain.

Based on these values, a formula to calculate the effort (*Eff*) in terms of time spent (*TSpent*) and the features inspected (*Features*) is presented in Eq. (13). According to this formula, the lower the value, the lesser is the effort required to inspect features.

$$Eff = \frac{TSpent}{Features} \qquad (13)$$

The amount of sub-domains in each iteration does not represent a large distribution that justifies an additional analysis for drawing conclusions about the comparison of results. Hence, we decided to include the whole data set available to characterize the inspection activity, toward establishing baseline values for the required effort.

Fig. 9 shows the values of the effort formula applied to the dataset represented by the amount of 92 features inspected during 139 h and 28 min. It results in a global average value of ≈1 h and 31 min to inspect every feature. However, this global value does not consider the details in terms of proportion. Hence, we applied the descriptive statistics in order to obtain a detailed view on the results of the effort calculations considering all the sub-domains. The descriptive statistics shows that the dataset range from 43.96 (min. value) to 268.33 (max. value). If we do not consider the outlier values, the max value is 106.5, median value of 95.11, mean value of 115.34 with a standard deviation of 73.04, as shown in the boxplots in Fig. 10.

On the effort distribution (Fig. 9), sub-domains **Scheduling** and **Inventory** are the most relevant of the analyzed sample. Nevertheless, as we did not have any baseline for this data, a more deep analysis could not be performed with this result.

Regarding the effort distribution across the tasks, the following charts present the ratio for effort required to perform every inspection task in all sub-domains. For example, in Fig. 11, the *planning* task in sub-domain **Patient** required 40.5% of the total effort, while *preparation* consumed 12.2% of the total effort, *meeting* took 25.3%, *correction* took 13.5% and *validation* consumed 8.4% of the

Table 14
Effort for performing inspection on feature specifications.

Tasks	Sub-domains								
	1st iteration					2nd iteration			
	A	B	C	D	E	F	G	H	I
Planning	120	120	120	120	120	120	120	120	120
Preparation	36	200	130	201	96	157	163	112	156
Meeting	75	408	228	78	96	201	210	120	180
Correction	40	1520	480	377	472	460	272	70	360
Validation	25	95	53	40	27	40	40	30	40
Number of features	4	22	23	8	4	11	3	8	9
Number of non-conformities	4	33	31	20	8	13	3	7	18

Sub-domains: A – Patient, B – Reception, C – Doctor's office, D – Billing, E – Scheduling, F – Supply management, G – Inventory, H – Purchase management, I – Laboratory.

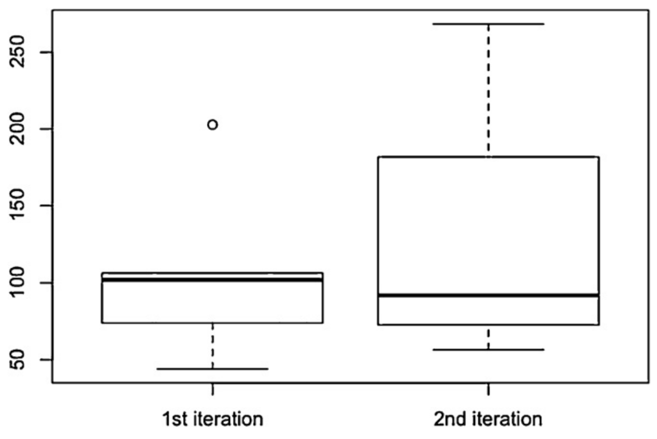


Fig. 10. Boxplot effort of inspection on features by iteration.

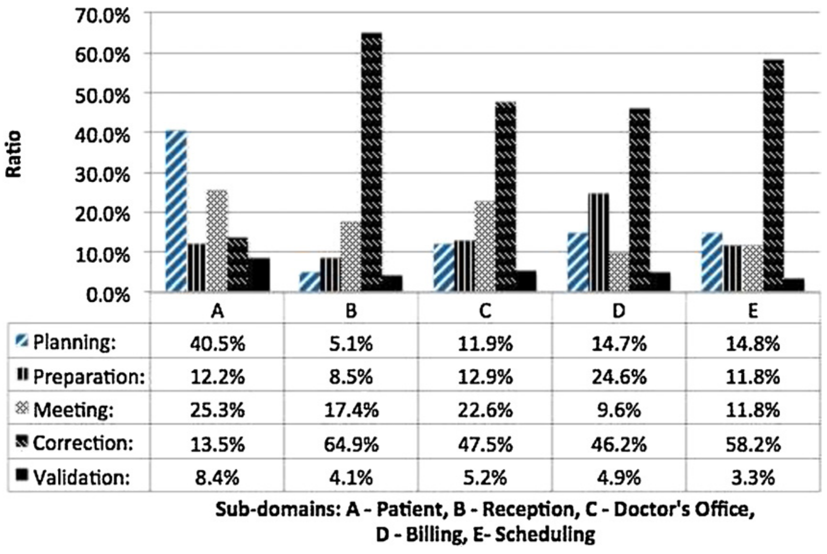
total effort. Figs. 11 and 12 show the detailed distribution considering all the tasks. On the other hand, Figs. 13 and 14 show the effort ratio, removing the *planning task* from the analysis, since we intended to analyze the behavior of the effort distribution considering floating values rather than the 120 min fixed planning task value, which could impact the global results.

According to the values presented in the four charts, there is a trend that the *Correction* task requires most effort. This could be identified in 7 out of 9 sub-domains. Nevertheless, the charts cannot give us confidence about which task requires more effort to be performed. More studies are necessary for building a baseline and investigating these issues.

8. Main findings of the study

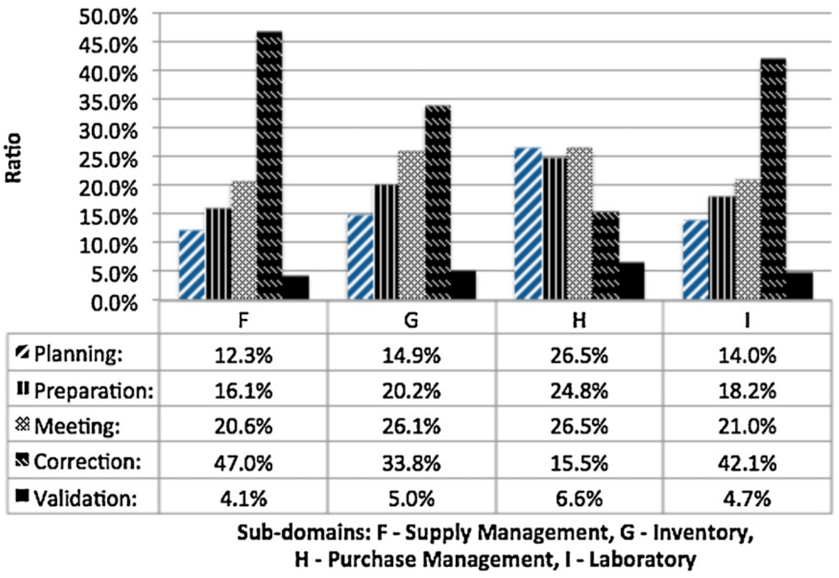
This study aimed to understand and characterize the software inspection activity on feature specification artifacts in an industrial SPL project. The objective was to gather evidence on the needed effort and the occurrence of non-conformities. The main findings of this study can be summarized as:

(a) The inspection activity reported *Incompleteness* as the main non-conformity type on feature specifications; and *Ambiguity* was highlighted as the second most common type of non-conformity. These findings can be attributed to the fact that the scope analysts did not have experience in building these artifacts in a SPL project.
(b) The *Business Rule* non-conformity type did not report any occurrence in this study. This can be explained by the high level of experience in the sub-domains, by the domain experts (Table 13).
(c) During the inspection work, the *Correction* inspection task was the most laborious. The *Correction* task required more effort in proportion to the sub-domains that have more non-conformities.
(d) The *Description* and Name template items reported most non-conformities for feature specifications.
(e) Optional features presented a higher non-conformity density when compared to mandatory features. The specified optional



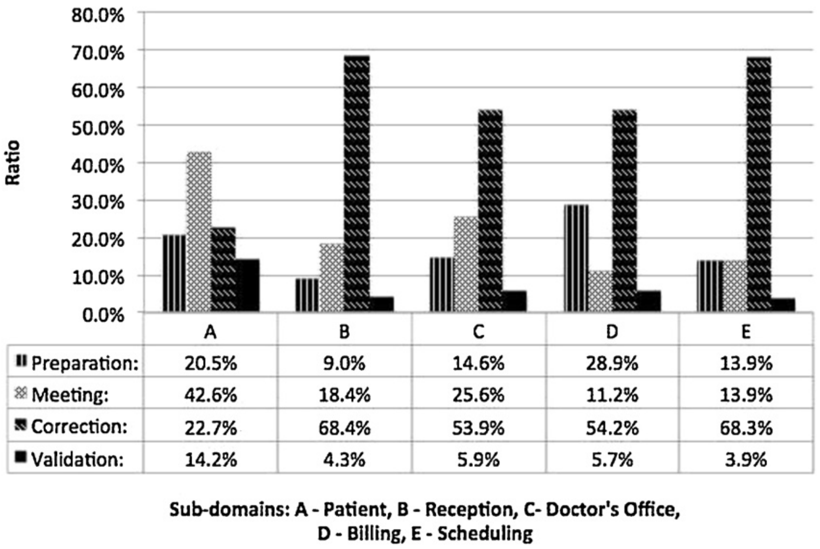
	A	B	C	D	E
▨ Planning:	40.5%	5.1%	11.9%	14.7%	14.8%
▮ Preparation:	12.2%	8.5%	12.9%	24.6%	11.8%
▩ Meeting:	25.3%	17.4%	22.6%	9.6%	11.8%
▧ Correction:	13.5%	64.9%	47.5%	46.2%	58.2%
■ Validation:	8.4%	4.1%	5.2%	4.9%	3.3%

Fig. 11. Effort distribution across feature inspection tasks – 1st iteration.



	F	G	H	I
▨ Planning:	12.3%	14.9%	26.5%	14.0%
▥ Preparation:	16.1%	20.2%	24.8%	18.2%
▩ Meeting:	20.6%	26.1%	26.5%	21.0%
▧ Correction:	47.0%	33.8%	15.5%	42.1%
■ Validation:	4.1%	5.0%	6.6%	4.7%

Fig. 12. Effort distribution across feature inspection tasks – 2nd iteration.



	A	B	C	D	E
▥ Preparation:	20.5%	9.0%	14.6%	28.9%	13.9%
▩ Meeting:	42.6%	18.4%	25.6%	11.2%	13.9%
▧ Correction:	22.7%	68.4%	53.9%	54.2%	68.3%
■ Validation:	14.2%	4.3%	5.9%	5.7%	3.9%

Fig. 13. Effort distribution across feature inspection tasks, without planning – 1st iteration.

features represent the variability concept in the SPL context, thus we already expected to have more number of non-conformities.

(f) Sub-domain attribute *Risk* allows building a predictive model (Poisson regression) for estimating non-conformities on feature specifications.

9. Lessons learned

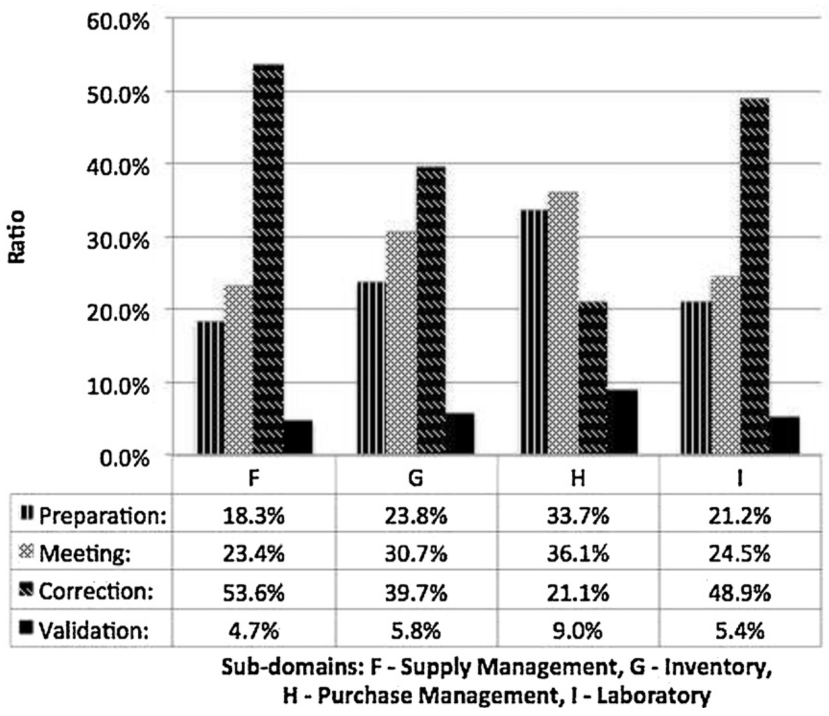
At the end of this empirical study, a workshop was held with all participants of the study in order to make a discussion forum about the inspection activity and to consolidate the lessons learned. Next, we describe the most important lessons learned that were collected:

Planning of inspection activities by sub-domain. In order to not waste effort in carrying out the inspection activity, and hence to promote the balancing of inspection tasks and, in particular, the task of inspection meeting, it was defined that the inspection meeting would be performed for one sub-domain at a time.

Duration of inspection meetings. In order to avoid boring inspection meetings and cause the participants to waste time, the inspection meetings should have a maximum duration of 2 h.

Using the product map and feature model as support materials to the inspection of features specification. During the inspection activity, we observed the need for using product maps and a graphical model for representing the interactions among features. In accordance with this observation, the participants of the inspection concluded that it was necessary to have a graphical feature model for better assessment of variability and dependencies among features. The use of a product map as a support was also mentioned as important.

Need to define a state machine for the specified features, to better manage and control the generated artifacts between inspection activities and the feature specification. The feature specification activity was supported by a tool that registered the features specified in the SPLSmart project. As the inspection activity dealt with sub-domains of the first and second iterations, some features appeared in more than one document generated by the tool. For the efficient management and control of the features



	F	G	H	I
▮ Preparation:	18.3%	23.8%	33.7%	21.2%
▨ Meeting:	23.4%	30.7%	36.1%	24.5%
▧ Correction:	53.6%	39.7%	21.1%	48.9%
■ Validation:	4.7%	5.8%	9.0%	5.4%

Fig. 14. Effort distribution across feature inspection tasks, without planning – 2nd iteration.

registered on the tool and reviewed on the inspection activity, we defined a state machine. The state machine has, initially, four states that a feature can assume: **specified**, **in inspection**, **inspected**, and **approved**.

10. Drawbacks and threats to validity

10.1. Drawbacks

During the study some drawbacks were identified:

- There were three issues that made the activities more difficult: (i) the lack of knowledge in the medical domain by the RiSE Labs members; (ii) the low involvement of the company staff during the adoption of the RiPLE process; and (iii) the low involvement of the company staff during the adoption of the software inspection approach. The RiSE Labs team was responsible for executing the SPL scoping through discussions with domain experts, since RiSE members are not experts in the domain. Thus, some time was spent in order to attend training and understand the medical domain.
- Due to the domain size and complexity, the management and tracking of variation points caused several problems, such as the management of asset corrections and evolution. In order to mitigate it, RiSE Labs developed a tool (Cavalcanti et al., 2011b) for managing the traceability among different SPL assets, such as: features, requirements, use cases and so on.
- Due to the reuse practice and commonality concept from SPL, one feature specified for one sub-domain can be reused by any other sub-domains in the product line. Thus, in order to better handle these artifacts on the cycle of specification, inspection and correction, and avoid reworking in the inspection activity, we created a state machine for the specified artifacts. This state machine was implemented and integrated in the tool developed by RiSE Labs (Cavalcanti et al., 2011b).

10.2. Threats to validity

There are some threats to the validity of the study, which we briefly describe next, along with the mitigation strategy for each one:

- **Observation:** The study was performed by observing the members of the inspection team and the artifact authors during the inspection meeting. However, the most important stakeholders might not have been invited to the inspection team. In order to mitigate this threat, some meetings were previously conducted in order to certify that the key stakeholders were invited to the inspection team.
- **Research Questions:** The set of questions might not have properly covered all the aspects of inspection in SPL. As it was considered a feasible threat, some discussions among the authors of this work and the members of RiSE Labs were conducted in order to calibrate the questions.
- **Classification of feature non-conformities:** The literature does not present any non-conformity taxonomy for feature specification artifacts. Thus, we adopted a taxonomy from a classification of requirement non-conformities (van Lamsweerde, 2009).
- **Negative results:** Some correlation analysis presented negative results. However, these should not be discarded immediately, and future analysis for these cases must be provided to increase the validity of the conclusion.
- **Reliability:** This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducted the same study, the result should be the same (Runeson and Höst, 2009). The reliability was achieved by using two tactics: a detailed empirical study protocol; and a structured study database with all relevant data such as meeting tapes, meeting spreadsheet files, transcripts, documents, and so on.
- **Meeting Timing Effect:** During the inspection meetings, due to the amount of features being discussed, it is possible that timing effects could lead the moderator to lose some information. In

order to avoid this threat, all meetings were audio-recorded, to be analyzed in details by the moderator later.

- **Confidence:** The potential bias introduced by the relationship between the researchers and the participants is important. In order to mitigate this threat, all data collection was treated strictly confidential, in order to protect company employees, who were part of the study, related to misuse of the information, and to ensure the complete freedom during the data collection procedures.
- **Generalization:** The findings and discussions were performed into the company context, which is characterized as a small company in terms of its software development team. Thus, although the findings and discussions could be generalized to larger companies, this cannot be guaranteed, since the challenges and problems can be completely different.

11. Conclusions

Software reuse is a key aspect for organizations interested in achieving improvements in productivity, quality and cost reduction. Software product lines, as a software reuse approach, have proven its benefits in different industrial environments and domains (Ahmed et al., 2007; Bastos et al., 2011).

Nevertheless, in order to achieve these benefits, quality issues should be considered, once each reused asset can be used in different products. The SPL literature has presented considerable efforts on SPL testing (Anselmo et al., 2010; Machado et al., 2011), with the development of approaches, tools and some empirical studies. However, the inspection area is not too rich in contributions (Babar et al., 2010), specially, with empirical studies based on industrial data. This paper allowed the initial characterization of the effort required to perform the inspection activity on the specification of features and indicated some trends, as well as quantified and characterized the occurrences of non-conformities in the context of a SPL project.

Based on the data, we identified that the **Correction activity** was the most laborious inspection activity, consuming more effort in 7 out of the 9 studied sub-domains. Regarding the occurrences of non-conformities, **Incompleteness** and **Ambiguity** presented the highest number of occurrences. For the template items of the specified features, **Description** had the largest number of non-conformity occurrences and the largest number of defective features. We identified also that the sub-domain risk information can be a good indicator for prioritization of sub-domains in the SPL inspection activity. In addition, the other identified findings can be used as baseline for further research.

This work can be seen as an initial step toward a precise guide for the inspection activity in SPL, and interesting directions remain to improve what was started here. As future work, we plan to replicate this study with another company in the financial domain and intend to investigate inspections in the SPL requirements engineering phase.

Acknowledgements

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[2]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

[2] http://www.ines.org.br.

References

Ahmed, F., Capretz, L., Sheikh, S., 2007. Institutionalization of software product line: an empirical investigation of key organizational factors. Journal of Systems and Software 80 (6), 836–849.

Anastasopoulos, M., Oliveira, T.H.B.D., Muthig, D., Almeida, E.S., Meira, S.R.D.L., 2009. Evolving a software product line reuse infrastructure: a configuration management solution. In: 3rd International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'09), pp. 19–28.

Anselmo, P., Machado, I.D.C., Cavalcanti, Y.C., Almeida, E.S.D., Garcia, V.C., Meira, S.R.D.L., 2010. A regression testing approach for software product lines architectures. In: 2010 Fourth Brazilian Symposium on Software Components Architectures and Reuse, pp. 41–50.

Babar, M.A., Chen, L., Shull, F., 2010. Managing variability in software product lines. IEEE Software 27 (3), 89–91, 94.

Balbino, M.S.M., Almeida, E.S., Meira, R.L.M., 2011. A scoping process for software product lines. In: 23rd International Conference on Software Engineering and Knowledge Engineering.

Bastos, J.F.S., Almeida, P.A.M., Meira, E.S.R.L.M., 2011. Adopting software product lines: a systematic mapping study. In: 15th International Conference on Evaluation and Assessment in Software Engineering.

Boehm, B., 1987. Industrial software metrics: top ten list. IEEE Software.

Burgos, T., 2008. RiPLE-EM: a process to manage evolution in software product lines. Master's Thesis. Federal University of Pernambuco, Recife, Pernambuco, Brazil.

Card, D.N., 1998. Learning from our mistakes with defect causal analysis. IEEE Software 15 (1), 56–63.

Cavalcanti, R.d.O., de Almeida, v, Meira, S.R., 2011a. Extending the riple-de process with quality attribute variability realization. In: Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS, New York, NY, USA. ACM, pp. 159–164.

Cavalcanti, Y.C., Do Carmo Machado, I., Da Mota, P.A., Neto, S., Lobato, L.L., De Almeida, E.S., De Lemos Meira, S.R., 2011b. Towards a software product lines metamodel to support variability and traceability. In: Proceedings of the 5th Workshop on Variability Modeling of Software Intensive Systems – VaMoS'11 (840), pp. 49–57.

Clements, P., Northrop, L., 2001. Software Product Lines: Practices and Patterns. Addison-Wesley, Boston, MA, USA.

da Mota Silveira Neto, P.A., 2010. A Regression Testing Approach for Software Product Lines Architectures: Selecting an Efficient and Effective Set of Test Cases. LAP Lambert Academic Publishing, Germany.

de Almeida, E.S., 2007. RiDE: the RiSE process for domain engineering. Ph.D. Thesis. CIn – Informatics Center, UFPE – Federal University of Pernambuco, Recife, PE, Brazil.

de Oliveira Cavalcanti, R., de Almeida, E.S., Meira, S.R.L., 2011. Extending the riple-de process with quality attribute variability realization. In: QoSA/ISARCS, pp. 159–164.

Denger, C., Kolb, R., 2006. Testing and inspecting reusable product line components: first empirical results. In: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, New York, NY, USA. ACM, pp. 184–193.

Fagan, M.E., 1976. Design and code inspections to reduce errors in program development. IBM Systems Journal 15 (3), 258–287.

Felty, A.P., Namojoshi, K.S., 2003. Feature specification and automated conflict detection. Transactions on Software Engineering and Methodology.

Filho, E.D.S., Cavalcanti, R.O., Neiva, D.F.S., Oliveira, T.H.B., Lisboa, L.B., Almeida, E.S., Meira, S.R.L., 2008. Evaluating domain design approaches using systematic review. In: Morrison, R., Balasubramaniam, D., Falkner, K.E. (Eds.), 2nd European Conference on Software Architecture (ECSA'08), vol. 5292 of Lecture Notes in Computer Science. Springer, pp. 50–65.

Gallegati, F.C.M., 2008. Paretos Law of Income Distribution: Evidence for Germany, the United Kingdom, and the United States. Microeconomics, Econ WPA.

Gautheir, T., 2001. Detecting trends using Spearmans rank correlation coefficient. Environmental Forensics 2 (4), 359–362.

Gilb, T., Graham, D., 1993. Software Inspection. Addison-Wesley, Boston, MA, USA.

Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S., 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21.

Karlström, D., Runeson, P., 2006. Integrating agile software development into stage-gate managed product development. Empirical Software Engineering 11, 203–225.

Khoshgoftaar, T.M., Gao, K., Szabo, R.M., 2001. An Application of Zero-Inflated Poisson Regression for Software Fault Prediction. Number 561. IEEE Computer Society.

Kollanus, S., Koskinen, J., 2009. Survey of software inspection research. Open Software Engineering Journal 3 (1), 15–34.

Lobato, L.L., O'Leary, P., de Almeida, E.S., de Lemos Meira, S.R., 2010. The importance of documentation, design and reuse in risk management for SPL. In: Proceedings of the 28th ACM International Conference on Design of Communication, SIGDOC'10, New York, NY, USA. ACM, pp. 143–150.

Machado, I.C., Mota Silveira Neto, P.A., Almeida, E.S., Lemos Meira, S.R., 2011. Riple-te: A process for testing software product lines. In: Proceedings of the Twenty-Third International Conference on Software Engineering and Knowledge Engineering – SEKE, pp. 711–716.

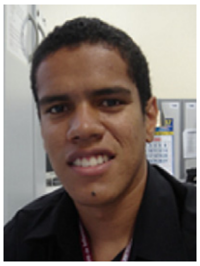
Maßen, T.v.d., Lichter, H., 2004. Deficiencies in feature models. In: Workshop on Software Variability Managment for Product Derivation – Towards tool Support in Conjunction SPLC 2004.

Montgomery, D.C., Runger, G.C., 2006. Applied Statistics and Probability for Engineers. Wiley.
Neiva, D., de Almeida, F., de Almeida, E., Meira, S., 2010. A requirements engineering process for software product lines. In: IEEE International Conference on Information Reuse and Integration (IRI), pp. 266–269.
Parnas, D., 2004. Software Inspections We Can Trust. Springer-Verlag, pp. 153–154.
Pezze, M., Young, M., 2007. Software Testing and Analysis: Process, Principles and Techniques. Wiley.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14, 131–164.
Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25, 557–572.
Segura, S., Hierons, R.M., Benavides, D., Ruiz-Corts, A., 2011. Automated metamorphic testing on the analyses of feature models. Information and Software Technology 53 (3), 91–105.
Shull, F., Singer, J., Sjøberg, D.I., 2007. Guide to Advanced Empirical Software Engineering. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
Thelin, T., Petersson, H., Runeson, P., Wohlin, C., 2004. Applying sampling to improve software inspections. Journal of Systems and Software 73 (2), 12.
Tian, J., 2001. Quality Assurance Alternatives and Techniques: A Defect-Based Survey and Analysis. Software Quality Professional 03.
van Lamsweerde, A., 2009. Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley.
Wainer, J. (2007). Methods of quantitative and qualitative research for computer science (in portuguese). Kowaltowski TB Karin editor Atualizaes em informtica, pp. 1–42.
Wallace, C., Cook, C., Summet, J., Burnett, M., 2002. Assertions in end-user software engineering: a think-aloud study. In: IEEE 2002 Symposia on Human Centric Computing Languages and Environments, pp. 63–65.
Wheeler, D.A., Brykczynski, B., Meeson, R.N., 1996. Software Inspection: An Industry Best Practice. IEEE.
Wiegers, K., 2002. Peer Reviews in software: A Practical Guide. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
Yin, R.K., 2008. Case Study Research: Design and Methods. Sage Publications.
Yourdon, E., 1989. Structured Walkthroughs, 4th edition. Yourdon Press.
Zhang, H., Kim, S., 2010. Monitoring software quality evolution for defects. IEEE Software 27 (4), 58–64.
Zhang, Z., Jiang, J., Liu, X., Lau, R., Wang, H., Zhang, R., 2010. A real time hybrid pattern matching scheme for stock time series. Reproduction 103 (January).

Iuri Santos Souza has a Bachelor of Information Systems degree from State University of Bahia (UNEB), Specialist in Software Engineering from Federal University of Pernambuco (UFPE) and a Master of Science degree in computer science (software engineering) from Federal University of Pernambuco (UFPE). Nowadays, he is member of the Reuse in Software Engineering (RiSE) Group, which has executed research regarding to Software Product Lines (SPL) Inspection. He is also participating on important research projects in Software Engineering area, as the National Institute of Science and Technology for Software Engineering (I.N.E.S.)

Gecynalda Soares da Silva Gomes is an assistant professor in the statistical area on Math Institute at Federal University of Bahia, Brazil. She received her PhD and MSc in computer science at Federal University of Pernambuco, Brazil.

Paulo Anselmo da Mota Silveira Neto was born in Recife, Pernambuco, Brazil. He has a Bachelor of Computer Science degree from Catholic University of Pernambuco (UNICAP), Specialist in Software Engineering from University of Pernambuco (UPE) and a Master of Science degree in computer science (software engineering) from Federal University of Pernambuco (UFPE). Nowadays, he is member of the Reuse in Software Engineering (RiSE) Group, which has executed research regarding to Software Product Lines (SPL) Testing, SPL Architecture Evaluation, Test Selection Techniques and Regression Testing. He is also participating on important research projects in software engineering area, as the National Institute of Science and Technology for Software Engineering (I.N.E.S.).



Ivan do Carmo Machado has a Bachelor of Information Systems degree from State University of Bahia (UNEB), Specialist in Software Engineering from Federal University of Pernambuco (UFPE) and a Master of Science degree in Computer Science (Software Engineering) from Federal University of Pernambuco (UFPE). Nowadays, he is student on PhD Computer Science program at Federal University of Bahia. He is member of the Reuse in Software Engineering (RiSE) Group, which has executed research regarding to Software Product Lines (SPL) Testing. He is also participating on important research projects in Software Engineering area, as the National Institute of Science and Technology for Software Engineering (I.N.E.S.).

Eduardo Santana de Almeida is an assistant professor in the software engineering area at Federal University of Bahia, Brazil, where he is member of Software Engineering Lab (LES) and head of the Reuse in Software Engineering (RiSE) Labs. He had studied in Virginia Tech to Post Doctoral, received his PhD in Computer Science at Federal University of Pernambuco, Brazil and the MSc at Federal University of São Carlos, Brazil. His research areas include: software reuse: methods, processes, tools and environments.

Silvio Romero de Lemos Meira is the chief scientist of Recife Center for Advanced Studies and Systems (C.E.S.A.R.). He is also a full professor of software engineering in the Informatics Center at the Federal University of Pernambuco. Born in city of Taperoá, state of Paraíba, Brazil. He holds a bachelor's degree in Electronic Engineering from the Aeronautical Institute of Technology (1977), a MSc and PhD degrees in computer science, respectively from the Federal University of Pernambuco (1981) and the University of Kent at Canterbury (1985). He is a software evangelist and has authored hundreds of scientific papers in the main journals and conferences in the software area, as well as presented a hundred of lectures around the world. He has also supervised a number of MSc and PhD students. His research interests involve a dozen of aspects on software engineering and correlated areas.