Information and Software Technology 67 (2015) 128–158

Contents lists available at ScienceDirect

Information and Software Technology

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

A systematic literature review of use case specifications research

Saurabh Tiwari [*], Atul Gupta

Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India

ARTICLE INFO

Article history:
Received 22 September 2014
Received in revised form 2 February 2015
Accepted 14 June 2015
Available online 23 June 2015

Keywords:
Use case specifications
Use case templates
Guidelines
Systematic reviews
Evolution
Quality

ABSTRACT

Context: Use cases have been widely accepted and acknowledged as a specification tool for specifying the functional requirements of a software system. Many variations of use cases exist which tries to address the issues such as their completeness, degree of formalism, automated information extraction, usability, and pertinence.
Objective: The aim of this systematic review is to examine the existing literature for the evolution of the use cases, their applications, quality assessments, open issues, and the future directions.
Method: We perform keyword-based extensive search to identify the relevant studies related to use case specifications research reported in journal articles, conference papers, workshop papers, bulletins and book chapters.
Results: The specified search process resulted 119 papers, which were published between 1992 and February 2014. This included, 54 journal articles, 42 conference papers, 2 ACM/IEEE bulletins, 12 book chapters, 6 workshop papers and 3 white papers. We found that as many as twenty use case templates have been proposed and applied for various software specification problems ranging from informal descriptions with paragraph-style text to more formal keyword-oriented templates.
Conclusion: Use cases have been evolved from initial plain, semi-formal textual descriptions to a more formal template structure facilitating automated information extraction in various software development life cycle activities such as requirement documentation, requirement analysis, requirement validation, domain modeling, test case generation, planning and estimation, and maintenance. The issues that remain to be sorted out are (1) the right degree of formalism, (2) the efficient change management, (3) the industrial relevance, and (4) assessment of the quality of the specification. Additionally, its synergy with other software models that are used in the development processes is an issue that needs to be addressed.

1. Introduction

Software Engineering is about developing software that includes requirement documentation, design principles, and other artifacts required to make the software function as per the stakeholders' expectations [117,133]. Software Requirement Specification (SRS) is a detailed description of the behavior of a system to be developed which includes the set of requirements describing user interactions with the system. There are various ways in which these requirements can be specified and analyzed that includes use case models, user stories, activity diagrams and formal languages [61,94,95].

Use case modeling is one of the specification documentation strategies that facilitates the developer to specify the intended user functionalities. Jacobson et al. [78] introduced the concept of use cases that have later been adopted by Unified Modeling Language (UML) [134]. The purpose of use cases is to delineate the requirements such that people without high-level training (e.g. the stakeholders, users) can understand and review them. It usually consists of two parts – use case diagram and the use case textual descriptions. These two parts depict two different sentiments of the specification, the first shows the structural representation of the use cases, actors and the relations among them, and the second presents the behavioral representation of the use cases using structured text written in natural language.

Use case models are typically used as the input in various software development activities, so their ability to clearly document correct, coherent, and an understandable set of functional requirements is critically important for the quality of resulting software product. Use cases have gradually become a widely accepted requirement specification technique both in research domain [2,10,13,32,33,74,103,151] as well as, to some extent, in

* Corresponding author.
E-mail addresses: saurabh.tiwari@iiitdmj.ac.in (S. Tiwari), atul@iiitdmj.ac.in (A. Gupta).

http://dx.doi.org/10.1016/j.infsof.2015.06.004
0950-5849/© 2015 Elsevier B.V. All rights reserved.

industrial practice [8,9,39,59,109]. Nevertheless, their inherent utilization of natural language and a behavior of requirement documentation in a semi-conventional way, mean that they are affected by issues such as ambiguity, redundancy, inconsistency, and incompleteness [2,32,146]. Several efforts have been made by researchers in order to address these issues by formalizing both behavioral and structural aspects of the use case specifications [16,57,69,141,151].

The literature on use case specifications provides a wealth of information on how to draw out, document, verify and validate the problem specifications [2,3,14,32,34,93,111,114]. This knowledge bank is presently distributed over numerous resources. The aim of this systematic review of the published literature on use case specification research is to collate knowledge such that use case developers have easy access to information needed to improve the effectiveness of use case specifications.

This paper reports the findings of a systematic literature review that focuses on the use case specification research works published between 1992 and February 2014. A total of 119 studies were identified from the 'hits' of a two-stage keyword-based search. We frame four research questions to investigate use case specification evolution, applicability, quality evaluation techniques, open issues and the future directions. We found that as many as twenty use case templates has been proposed and applied in different applicability contexts ranging from informal textual descriptions with paragraph-style, keyword-oriented format to a more conventional flair. However, we found just one study evaluating the usability aspects of various use case templates that has been carried out in an academic context, and this suggests a need for investigating the same in both industrial and academic settings. We identified various quality assessment techniques such as rules, guidelines, checklists and quality assessment attributes used to specify and validate the use case specification, and as a result, we highlighted their strengths and weaknesses.

The outcomes of the study highlight that the inherent variability of use cases for which to document the specifications raises several questions regarding their comprehensibility, learnability, maintainability, traceability, completeness, correctness, right degree of formalism, and applicability in developing other software artifacts during the software development process. Moreover, we found very few studies carried out in industrial domain for investigating the applicability of the use cases in different contexts, thus we could not able to substantiate the claims more widely. However, from the empirical studies carried out in the academic domain, one can make out that the strength of the evidences to support the claims are acceptable and interesting. But, in order to substantiate further, the similar studies need to be carried out in the industrial domain. Finally, we conclude that there is a demand for more rigorous research on improving the quality of use case specifications in various software development activities to establish a communication between the stakeholders associated with the system.

The organization of the rest of the paper is as follows. Section 2 presents the works related to our study and accordingly, outlines the contributions of the paper. Section 3 describes the research methodology adopted in this study. Section 4 presents the results and findings of the selected studies. Section 5 enumerates the research findings. Section 6 discusses the implications of our study along with discussions on the threats to validity, while Section 7 concludes and suggests future work.

2. Related work

There are few systematic reviews reported in this research area. El-Attar et al. [49] presented a systematic literature review of the present practices that were typically used and applied to develop quality use case models. In their study, they unified all the information/syntactical rules typically used for developing use case models that may be misapplied, resulting generation of low quality use cases and other derived software artifacts. To this end they recommended 21 anti-patterns as per [45,48]. Hurlbut et al. [68] presented the results of a survey conducted to identify and analyze the approaches available in the literature to describe and formalize the use cases. The major elements of the use case description and formalization approaches are classified in two different perspectives – first, use case representation (the categories are textual, graphical and dynamic) and second, use case focus (the categories are static, dynamic, policy, and process). The result of the survey showed the overview of use cases in both the perspectives – focus and representation – similar to the systematic mapping studies.

Even though use cases are both a very well-known and widely used software specification technique (i.e., used in a very heterogeneous and diverse ways), there has not been any significant study in whole, targeting more comprehensive details about the use cases. For instance, El-Attar et al. [49] study focused only on to the guidelines, suggestions and techniques provided to develop quality use case specification. Hurlbut et al. [68] surveyed the use case literature from 1992 to 1997 which is periodically a time when UML adopted the use cases, thereafter a lot of researches have been conducted for applying them in different contexts. Furthermore, there exist many studies for investigating the quality of use case specification by possessing the specific syntax, pitfalls, recommendations, notations, syntactical rules or semantics [2,15,49,114,115], but, there have not been studies on whether such mechanisms can establish their relevance in practice.

Our systematic review targeted on identifying the open issues and future directions that need to be evaluated in order to improve the effectiveness of the use case specifications based on the related relevant studies published between 1992 and February 2014. We have also consolidated the commonly used quality measures to specify, validate and analyze the use case specification. For this, we formulate four research questions – use case evolution, applicability, quality and the issues/future aspects – in order to summarize the significant information (i.e., used in industry or research or academia) disseminated across numerous literature resources into a single resource.

3. Research methodology

In this paper, we adopt the methodology used for conducting a systematic literature review recommended by Kitchenham et al. [88–90]. Some parts of the methodology used in the study have been presented by other systematic reviews [1,42,43]. Fig. 1 shows an overview of the phases involved in the review study.

In the first phase, we formulated the research questions. In the second phase, we selected the search terms and the literature resources. The third phase dealt with the selection of the papers, while the fourth phase refined the selection by a standardized quality assessment process. And last, the fifth phase involved data analysis.

3.1. Research questions

The aim of this systematic review is to examine the existing literature on use case specifications research to identify (1) how use cases have evolved and how are they used to specify the problem requirements (2) their applicability in various software development activities (3) their quality (4) the open issues and the future directions. We frame four research questions that are as follows:

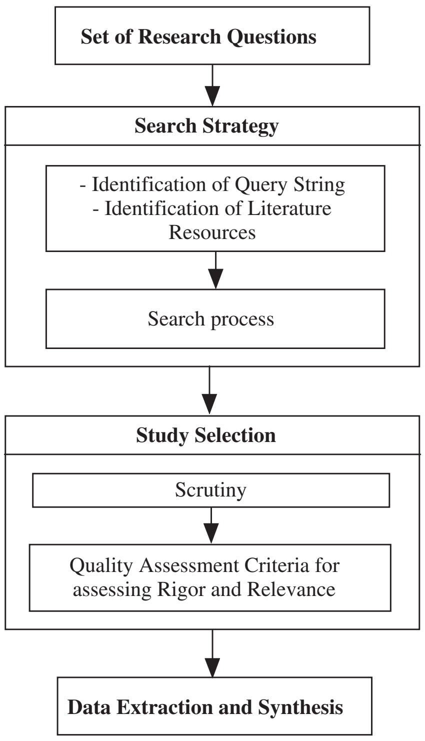
Set of Research Questions
Search Strategy
- Identification of Query String
- Identification of Literature Resources
Search process
Study Selection
Scrutiny
Quality Assessment Criteria for assessing Rigor and Relevance
Data Extraction and Synthesis

Fig. 1. Overview of the phases involved in the review study.

RQ1 How have use cases evolved and which use case template is most useful in practice?
RQ2 How have use cases been used in various software development life cycle activities?
RQ3 How the quality of use case specifications can be assessed?
RQ4 What are the open issues and the future directions of the use case specifications research?

As the research questions are not independent, they were intertwined and simultaneously investigated (see Fig. 2).

3.2. Search strategy

In this section, we present a detailed description of the search strategies used to identify the relevant studies on use case specification research. The search strategy consists of the search terms, the literature resources and the search process. We started our search by keying out the set of query strings to perform the searches from the literature resources.

Before starting the review process, we pilot the search to find relevant research papers by performing an initial search in seven Journals and three Conferences namely, IEEE Transactions on Software Engineering, Information and Software Technology, Requirements Engineering Journal, Journal of System and Software Modeling, Software and Systems Modeling, Software Quality Journal, ACM/IEEE International Conference on Software Engineering, IEEE International Conference on Model Driven Engineering Languages and Systems, International Conference in Software Engineering, and IEEE International Requirements Engineering Conference. The pilot search was performed independently by two authors, and then the search results were collated in order to formulate/modify the search query strings, the inclusion and exclusion criteria, and the selection strategies.

3.2.1. Identification of query string

Established along the research questions, following steps were employed to build the search terms [75,89].

1. Filiations of the major terms of the research questions such as use case specifications, use case templates, quality of the use cases, and applicability of the use cases.
2. Recognition of the keywords in relevant papers and software engineering text books.
3. Recognition of some terms related to psychology, natural language, generation, automation, human intensions, quality, understanding, usability, formalism and linguistic analysis.
4. Usage of Boolean 'AND' and Boolean 'OR' to combine relevant search strings. (A Boolean 'AND' between two search strings indicate the concatenation of the two search string and a Boolean 'OR' between two search string indicates selecting either of the two search strings.)

We divided the set of search terms into two categories - General and Specific [75,150]. General terms are the keywords which are the major terms deduced from the more comprehensive range of the research questions. The specific terms consist of the keywords specific to the use cases, their quality assessment and some common terms that may bear upon the usage of use cases in software evolution process. This also includes the terms related to their use in automated generation of various software artifacts. The terms used to generate the query strings are as follows.

General Terms: Use Case; Use cases AND (diagram OR requirements OR specifications OR formalism OR textual descriptions OR templates OR models OR validation OR quality OR applicability OR evolution OR realization OR applications OR functional OR documentation OR analysis OR development OR generation OR construction).

Specific Terms: Use case; Use cases AND (requirements/OR descriptions/) AND (automation OR derivation OR evaluation OR verification OR validation OR generation OR model transformation OR linguistic OR natural language processing OR classification OR assessment OR rules OR guidelines OR checklist OR empirical OR experimental OR testing OR test cases OR domain analysis OR effort estimation OR planning OR validation OR maintenance OR change analysis OR impact OR safety analysis).

3.2.2. Literature resources

The six electronic databases used were: IEEE Xplore (IEEE),[1] ACM Digital Library (ACM),[2] ScienceDirect – Elsevier,[3] SpringerLink,[4] Scopus,[5] and Compendex (CPX).[6] We considered journal papers, conference proceedings, workshops, symposiums, and ACM/IEEE bulletins to conduct the searches. The technical/experience reports, white papers and the books chapters were searched by reviewing the references of the selected papers.

3.2.3. Search process

The initial search for finding the relevant papers revealed a total of 51 articles related to the use case specification research. These 51 articles were identified from the following sources.

- Requirements Engineering Journal (REJ) – 12 Papers.
- Information and Software Technology (IST) – 11 Papers.
- Empirical Software Engineering (ESE) – 5 Papers.
- IEEE Transactions on Software Engineering (TSE) – 4 Papers.
- Journal of Systems and Software (JSS) – 1 Paper.
- System and Software Modeling (SoSYM) – 5 Papers.
- Software Quality Journal (SQJ) – 2 Papers.

[1] www.ieeexplore.ieee.org.
[2] www.portal.acm.org.
[3] www.sciencedirect.com.
[4] www.link.springer.com.
[5] www.scopus.com.
[6] www.engineeringvillage2.com.

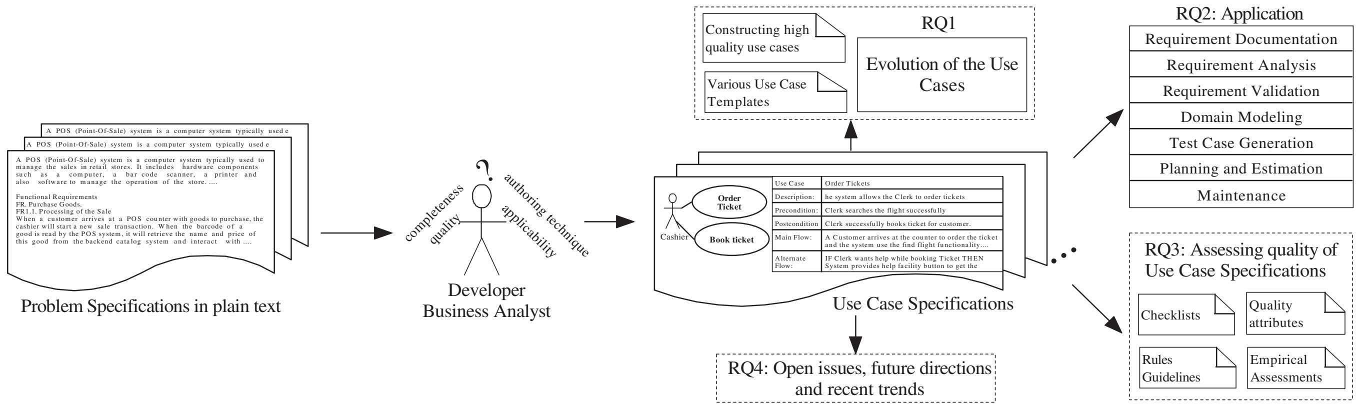

Fig. 2. Research questions.

- International Conference on Software Engineering (ICSE) – 2 Papers.
- IEEE International Requirements Engineering Conference (RE) – 6 Papers.
- IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS) – 3 Papers.

The electronic search process was conducted in two stages. First, the keyword-based search was launched in the six electronic databases. Relevant papers were flagged by analyzing the titles and abstracts. Any duplicate finds were removed. Second, the reference and citation lists of all relevant papers (identified in first stage of selection) were then analyzed to find additional papers and if identified, were added to the ones identified in the first stage. The overview of the two-stage search and selection process along with the studies identified at each level is shown in Fig. 3.

3.3. Inclusion and exclusion criteria

The systematic literature review conducted in this paper aimed to identify the studies targeting the use of use case specifications in various software development activities between 1992 and February 2014. The studies which provide relevant contributions for specifying, validating and improving the quality of use case specifications have been selected. For this, we have included the empirical studies, Solution Proposal papers and the opinion papers (experience/technical reports, bulletins and white papers) in our study. The papers which do not consist of sufficient technical details about the use case specifications research were excluded. Only studies written and published in English language from peer-reviewed journals, refereed conference proceedings, workshops, symposiums, book chapters and ACM/IEEE bulletins were considered for inclusion in the list of relevant studies.

A summary of the criteria used for scrutiny are summarized as follows.

Inclusion Criteria

– All papers published in English language.
– Papers that focus on constructing high quality use case specifications.
– Published between 1992 and February 2014.
– Papers that include information about use case specification specification (i.e., the rules, the guidelines, the checklist, the templates, and their applications).
– All published papers that have the potential of answering at least one research question.
– Papers that make use of use case specifications to generate other software artifacts (e.g. domain analysis models, test cases, etc.).

Exclusion Criteria

– Papers that are not published in English language.
– Papers that do not have any link with the research questions.
– Prefaces, interviews, reviews, posters, panel discussions, tutorial summaries, and article summaries.
– Duplicate papers (only the most complete, recent and improved one is included).
– Papers without bibliographic information such as publication date/type, volume and issue numbers.
– Papers that only provide a use case model without discussing its relevance or notations.
– Papers with inadequate information regarding utilization of the use cases in various software development activities.

3.4. Study selection

In the two-stage study selection process, we applied inclusion and exclusion criteria to select the studies. From the first stage process, a total of 1289 papers were identified. For each selected paper, their subsequent titles and abstracts were scrutinized to determine whether it belongs to our present study. If the title and abstract could not assist us to derive a conclusion, then we checked the paper's full text. This task was necessary to eliminate duplicate and irrelevant studies. Consequently, 205 relevant papers were selected. Thereafter, an automated search based on citation analysis (a.k.a. forward snowballing) was performed by searching all papers referring these 205 papers. Also, the references of each selected study were scrutinized (a.k.a. backward snowballing) to identify important studies that might have been missed out during the initial search process. This effort led to the identification of 112 additional studies (i.e., 23 from forward snowballing and 98 from backward snowballing). Finally, the quality assessment criteria were applied to these 317 studies. This lead to a final selection of 119 studies for the data extraction and synthesis.

3.4.1. Scrutiny

Fig. 3 depicts the two-stage selection process for identifying relevant studies from the identified ones. A total of 1289 papers were identified by searching the studies on the potentially relevant literature resources. Next, we scrutinize each papers in order to remove the duplicates studies. We have checked the title and abstract of the papers to determine whether study belongs to the formulated research questions. If we both the authors were unable to draw some conclusions, then we go through the paper's full text. 58 paper's full text was checked completely as the title and abstract of the paper could not help us to make a decision. Out of these 58 papers, this resulted in the identification of 15 relevant papers.

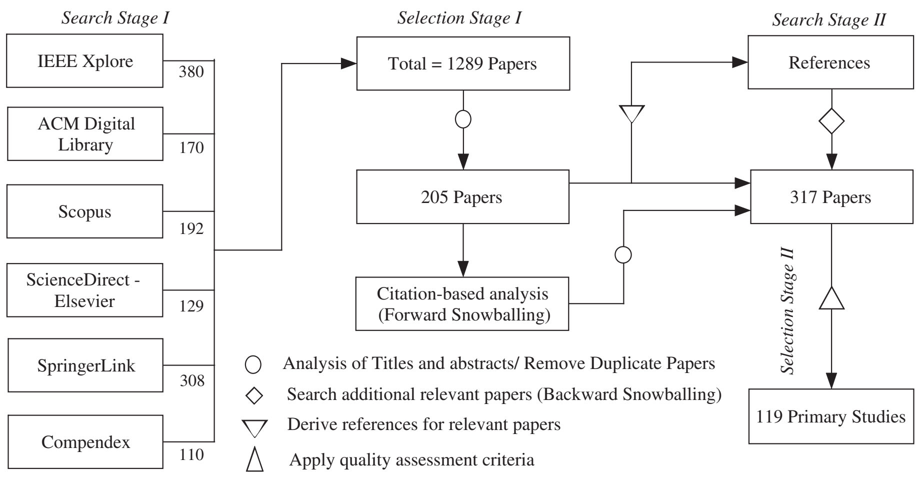
Search Stage I
IEEE Xplore
380
ACM Digital Library
170
Scopus
192
ScienceDirect - Elsevier
129
SpringerLink
308
Compendex
110
Selection Stage I
Total = 1289 Papers
205 Papers
Citation-based analysis (Forward Snowballing)
Search Stage II
References
317 Papers
Selection Stage II
119 Primary Studies
○ Analysis of Titles and abstracts/ Remove Duplicate Papers
◇ Search additional relevant papers (Backward Snowballing)
▽ Derive references for relevant papers
△ Apply quality assessment criteria

Fig. 3. Search and selection process.

This process of scrutinizing the papers were independently conducted by each author and then the results were collated and any disagreements between the authors were discussed to reach final consensus.

We have excluded the papers which were not (1) able to address any of the formulated research questions (2) written/published in English language, and (3) sufficient technical information regarding use case specification proposals. However, if the multiple copies exist then we include the recent, complete and improved one (paper's full text were checked, wherever applicable). More importantly, the included paper were published between 1992 and February 2014. A summary of criteria used for the scrutiny is shown in Section 3.3.

3.4.2. Quality assessment

Our primary studies were of different types: Empirical Studies, Solution Proposals and Opinion Papers. In order to assess the quality of selected studies, we classified the 119 studies into four different categories suggested by [110,147] and then used a set of quality assessment questions for each category as suggested by [41,43,147].

The four different categories used for classifying the primary studies are shown below.

1. *Evaluation Research Papers* – This represents the techniques that are implemented in practice and rigorous evaluation is conducted.
2. *Validation Research Papers* – This represents the papers that include the evaluation of a technique using some example experiments, i.e., work done in the lab. These types of investigations are novel and yet not been implemented in practice.
3. *Solution Proposal Papers* – This represents a solution to a problem, that can either be novel or an extension of an existing use case specification technique. The applicability and strengths of the solution are demonstrated by an illustrative example or case study or argumentation.
4. *Opinion Papers* – This represents the researchers' opinions/experiences about the use case specification technique, highlighting their strengths and weaknesses. These types of studies do not rely on the related works and the research methodologies.

The quality assessment (QA) of selected studies was done by scoring the set of questions specified in different categories so as to obtain relevant studies capable of addressing at least one research question. For this, we formulated a number of quality assessment questions (see Table 1) to evaluate the quality of the selected studies. Each question is judged against three possible answers: 'Yes' (score = 1), 'Partly' (score = 0.5) or 'No' (score = 0) [76,85]. Consequently, the 'quality score' for a particular study is computed by taking the sum of the scores of the answers to all the questions. The judgment process was carried out in two steps. In step one, both the authors separately judged the answers of these questions. Next, the two authors shared their answers, discussed the rationales and finalized the answers.

Subsequently, we have included only those studies as primary, whose 'quality score' should be greater than 0.5 on the scale of 0–1. Specifically, greater than 7 for validation and evaluation research papers, greater than 4 for solution proposal papers, and greater than 3 for opinion papers. As a result, 198 papers were excluded from the initially collated studies giving rise to 119 finally selected relevant studies (see Fig. 3). The quality scores of these selected studies are presented in Tables A.20 and A.21. These metrics help to interpret and assess the strength of evidence (i.e., rigor and relevance) of each study included in the systematic review.

3.5. Data extraction and synthesis

The purpose of data synthesis is to scrutinize the selected papers in order to answer the research questions. The papers selected in the study were the journal articles, conference papers, ACM/IEEE bulletins and book chapters, where the findings have been supported by conducting the case studies, surveys, experimental evaluations, empirical assessments, observations or questionnaires. To extract the detailed information from the 119 selected studies, a data extraction form were used to summarize proofs from the selected primary studies w.r.to the criteria defined, as shown in Table 2. Purposefully, this form was designed to collect data from the primary studies needed to answer the formulated research questions.

The data extraction form shown in Table 2 included the following properties such as *study detail*, *study description*, *detailed assessments*, and *study findings*. The *study detail* property include basic details of the studies such as author's name, paper's title, publication details and publication venue. The *study description* property include details about the research type, aim, relevance, and the application context. The *detailed assessment* property include the detailed information about the methodology applied to show case the usefulness of the use case specification research. And, the *study findings* property include the subjective assessment of the studies highlighting their outcomes, strength, weakness, future scopes;

Table 1
Quality Assessment Questions [41,90,147].

Q. What research method was used is this? – Validation Research, Evaluation Research, Solution Proposals and Opinion Papers (technical/experience reports, white papers, bulletins)
Note: This was based on our reading/understanding of the paper not the method/approach claimed by the author(s) of the paper.

Category	Quality assessment questions
Evaluation/ Validation Research	QA1. Is the problem clearly stated?
	QA2.Was the research method appropriate to address the aims of research?
	QA3.Is the knowledge claim validated (i.e., Is the conclusion supported by the paper)?
	QA4.Is the technique/template/specification to be validated clearly described?
	QA5.Is the research significantly increase the knowledge about use case specification research?
	QA6.Is the broader relevance of the work discussed?
	QA7.Is there an adequate description of the context in which the research was carried out so that the practitioners can validate/replicate it in later research?
	QA8.Was the data collected in a way that addressed the research issue?
	QA9. Is there a clear statement of findings?
	QA10. Was the recruitment strategy (for human-based experiments and quasi-experiments) or experimental material appropriate to the aims of the research, if applicable?
	QA11. Was there any control group present with which the treatments can be compared, if applicable?
	QA12. Was the data analysis sufficiently rigorous?
	QA13. Has the relationship between the researcher and participants been considered to an adequate degree, if applicable?
Solution Proposals	QA1. Is the problem to be solved by the technique clearly explained?
	QA2.Is the technique novel (or Is the application of the techniques to this kind of problem novel)?
	QA3.Is the technique sufficiently well described?
	QA4.Is the technique sound?
	QA5. Is the broader relevance of this technique argued?
	QA6. Is it clear under which circumstances the technique has the stated properties?
	QA7. Is there sufficient discussion of related work? (Are competing techniques discussed and compared with the present technique?)
Opinion Papers	QA1. Is there a clear statement of the aims of the study?
	QA2. Is there an adequate description of the context in which the observations was carried out?
	QA3. Are the lessons learned interesting?
	QA4. Is the article revealing?
	QA5. Is the article relevant for practitioners?

and more importantly, the study belongs to which research question – RQ1 or RQ2 or RQ3. There might be some studies that fall under one or more research question, thus analyzed accordingly.

The studies belonging to a specific research question were then elaborated as per the information articulated in the findings and conclusion section. This resulted in the identification of the issues/findings/aspects that have been well-undertaken, and the issues that need to be addressed/explored/evaluated in the future to improve the quality of use case specification technique. The preliminary details of each study (e.g., study detail and study description) was collected by one of the author; and the study findings by both the authors in collaboration. Both the authors meticulously extracted the contents from the study in order to draw the conclusions. All discrepancies on the outcomes/results/analysis were discussed among the authors with the aim of reaching consensus.

Table 2
Data extraction form.

(I) Study detail:	
Paper ID	Unique ID of a study
Authors	Name of the authors
Title	Title of the paper
Publication venue	Journal, conference papers, bulletins, book chapters, workshops, white papers, symposium
Details of publication	Date, Year, Page Numbers
(II) Study description:	
Research methodology	Experimental, Empirical, Observational, Tool Development, Solution Proposals, Experience Reports, Others
Aim of the study/study focus	Domain topic, problem solution, replication, conceptual framework, validation, observations, model transformation, evaluation
Relevance	Research, Practice
Applied settings	The settings in which the technique is applied/evaluated/observed
(III) Detailed assessment:	
Applicability context	Identification of the life cycle phases in which the study falls under
Setting of the study	Industry, academia, in-house, products, processes
Design of the study	Qualitative, Quantitative (experiment, survey, case study, action research)
Method proposed	Name of the method and their description
Validation/evaluation performed	Does the validation/evaluation performed in the study?
Research hypothesis, if any	Does the hypothesis formulated for the analysis?
Data collection	How was the data obtained?
Data analysis	How was the data analyzed?
(IV) Study findings:	
Findings and conclusion	The conclusions and findings that can be drawn
Observations	The observations drawn out by analyzing each primary study
Limitations/threats to validity/Constraints	Identification of the study's shortcomings and areas for future research
Mapping to the formluated RQs (RQ1 or RQ2 or RQ3)	Under which research question the primary study belongs?

The detailed assessment of each primary study was independently carried out by the authors of this paper. The identified results of both the authors were then collated together and disagreements were discussed among the authors to fill the relevant information in a data extraction form. After extracting data from the primary studies, we applied appropriate synthesis methods as suggested by [38], in order to analyze each study included in the systematic review. The synthesis methods applied for each of the studies belonging to a research question is shown in Table 3.

To collate the identified information with respect to RQ1, we have used tables, figures and other visual tools to show how use cases have evolved. The information related to RQ2 was organized in a paragraph and tabular style, where we attempt to answer the question w.r.to the applicability of use cases in various software development activities along with the basis of findings and the observations illustrated in the paper. In RQ3, we focused on the identification of quality measures, rules and the checklists proposed in the literature to improve the quality of use case specifications. The set of guidelines, checklist and the quality attributes were illustrated by documenting them in a tabular format, and, the experimental assessments conducted in the literature were shown in a tabular and paragraph style format. Additionally in RQ4, the overall results of the study were summarized by addressing the open issues, future directions and the recent trends in a paragraph style format.

Table 3
Data synthesis.

Synthesis method	RQ	Description
Narrative Synthesis	RQ1	For *RQ1*, the studies were observed in order to depict how use cases evolve
Thematic synthesis	RQ1, RQ2, RQ3	For *RQ1*, we identify the patterns within the data for analyzing, and reporting use case specification research within the studies For *RQ2*, we identify the set of areas where use cases are applicable for categorization of the studies in order to further analyze them for the synthesis For *RQ3*, we identify the patterns how the quality of the use case specifications be improved
Cross-case analysis	RQ1, RQ2, RQ3	For *RQ1*, we identify how use case templates have been evolved in the literature, The tables and the graphs were used for the analysis and interpretation. The differences and similarities between the use case templates stated in the studies were also depicted For *RQ2*, we categorize them into six categories according to their use in various software development activities (identified through the thematic analysis). Then, we analyze the studies based on tabular displays For *RQ3*, we categorize them into four categories, and consolidate the findings of the studies fall under this RQ is presenting using tabular displays
Content analysis	RQ2, RQ3	For *RQ2*, the frequency of occurring of a specific technique in particular category was counted and tabulated. The frequencies of data occurred in a specific category suggest the usefulness of the use case specification Similarly, for *RQ3*, the frequency of occurring of a technique was counted and tabulated to determine their strength and limitations

4. Results and analysis

This section discusses the findings of this systematic review. We begin by giving an overview of the selected works and then a detailed description of the findings of this review followed by findings pertinent to the research questions.

4.1. Overview of selected studies

119 studies were selected for this research. Among them, 54 were journal articles, 42 were conference papers, 2 papers came from ACM/IEEE bulletins, 12 were book chapters, 6 were workshop papers, and 3 were white papers. The percentage of the selected studies drawn from various resources are shown in Fig. 4, the overview of the studies belongs to each research question is shown in Table 4, and the number of papers used in the study by year of publication is depicted in Fig. 5. However, the description of the selected studies along with the publication venues shown in Table C.23 of the appendix section.

4.2. Quality of research

In this section, we discuss the quality of the selected primary studies based on their research and evaluation types.

4.2.1. Research type

Table 5 shows the statistics on the research type of study in the selected primary studies. We found that the majority of the use case specification researches involve solution proposals, and a total of 32 opinion papers that sketch a new way of looking at the use case specification and have not yet been implemented in practice.

As seen from Table 5, 21 studies have been conducted in academic settings using some example experiments/case studies, and 15 studies have been carried out in industrial settings or by applying industrial strength case studies. These represent a small number of studies in order to substantiate the claim. Hence, there is a need to conduct more empirical studies, both in academic and industrial settings to establish the relevance of the use cases in a specific context, rather than proposing new extensions of the use case modeling approaches.

The 32 primary studies that fall under the opinion papers category include the experience/technical reports, the white papers, the book chapters, the workshops and the bulletins. The distribution of the research type of these studies is shown in Fig. 6. From the figure, it can be depicted that a large amount of studies on improving the quality of the use case specification is presented in terms of the experience reports, the book chapters, and the white papers, which were illustrated either by using some toy examples or small case studies. Though, these studies are interesting and worth considerable, but lacks on the evaluation.

4.2.2. Evaluation type

In this section, we evaluate the relevance of the primary studies based on their evaluation types. For the first two categories – evaluation and validation research, we found very few studies that are conducted in industrial settings, hence no industrial relevance for the use case specification can be claimed. Apart from this, Fig. 7 depict the statistics on how solution proposal primary studies have been evaluated. We categorized the evaluation scheme into five types: (1) a case study on a small industrial system (2) industrial case study, which includes large projects with significant number of functionalities. (3) tool development (4) illustrative examples, and (5) academic case studies, which includes small student developed projects or examples illustrated in the textbooks.

As use case specifications have been applied in various software development processes, there exist several problems that must be solved before applying them in practice. For this, several proposals have been made and validated. Fig. 7, shows the evaluation types made to validate the solution proposals. From the total of 51 solution proposal articles, 35% are evaluated by applying the small industrial case studies, 18% of the studies are evaluated in larger industrial contexts. Furthermore, 14% studies develop a tool support for validating their proposals, 12% of the studies shows the proposal applicability using some illustrative examples and rest 21% of the studies are evaluated using academic projects. Arguably, one can see that 67% of the solution proposal studies are evaluated by applying industrial case studies (i.e., small or large scale systems) or by developing the tool supported approaches, which indicates the applicability and usefulness of the use case specifications on such contexts. But, in order to substantiate the claim more widely, specific to each context, more empirical studies need to be carried out (as we found a less number of studies lies in the validation and empirical research categories). In the next following sections, we explored four research questions on the basis of selected primary studies and a set of interesting findings related to them are summarized.

Analysis of these studies revealed that despite having several results obtained from the solution proposals and the empirical evaluations, there arises an issue when it comes to the industry to adopt

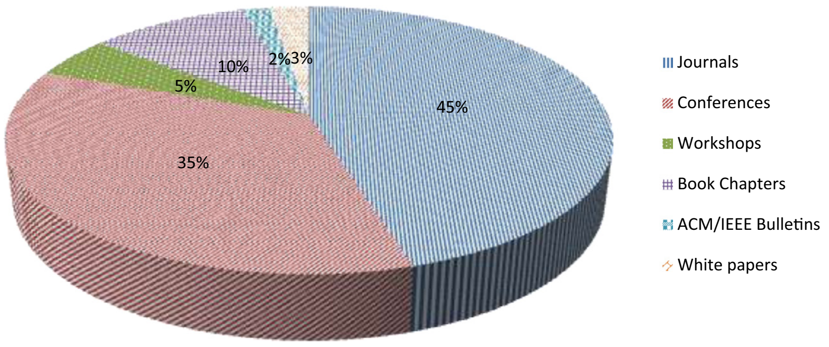

Fig. 4. Percentage of papers drawn from various resources.

Table 4
Overview of the studies belongs to each RQ.

RQ	Study
RQ1	[2,19,63,23,32,57,3,16,11,6,78,40,122,145,124,131,139,45,100,140,4,114,67,113,120,127,146,125,97,108,99,74,96,77,79,86,92–94,64]
RQ2	[14,129,148,137,56,132,144,118,145,124,122,80,40,105,121,135,141,143,149,73,87,128,131,130,123,126,102,104,106,107,119,82,83,100,98,46, 52–54,62,65,66,71,26,45,27,30,151,136,13,112,36,144,22,12,18,20,51,101,113,125,96,99,7,29,17,60,61,103,5,8]
RQ3	[10,25,118,48,58,144,31,128,126,45,44,151,139,55,115,112,81,37,47,50,35,36,33,64,97,22,15,12,2,11,4,39,101,116,138,142,114]

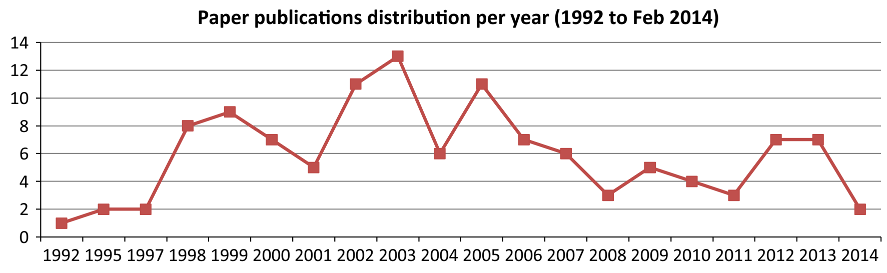

Fig. 5. Number of papers included in the study by the year of publication.

Table 5
Statistics on the research type of study.

Category	Studies	Count
Evaluation Research Papers	[2,4,14,18,20,39,51,101,116,138,142,7,29,103,114]	15
Validation Research Papers	[10,12,15,21,22,33,35–37,47,50,55,81,112,115,128,139,140,151,136,13]	21
Solution Proposal Papers	[25–27,30,44–46,52–54,62,65,66,71,78,82,83,98,100,102,104,106,107,119,123,126,130,131,135,141, 143,149,73,87,121,105,144,31,40,48,58,80,118,122,129,148,137,56,132,145,124]	51
Opinion Papers	[3,5,6,8,11,16,17,19,23,32,57,60,61,63,64,67,74,77,79,86,92–94,96,97,99,108,113,120,127,146,125]	32

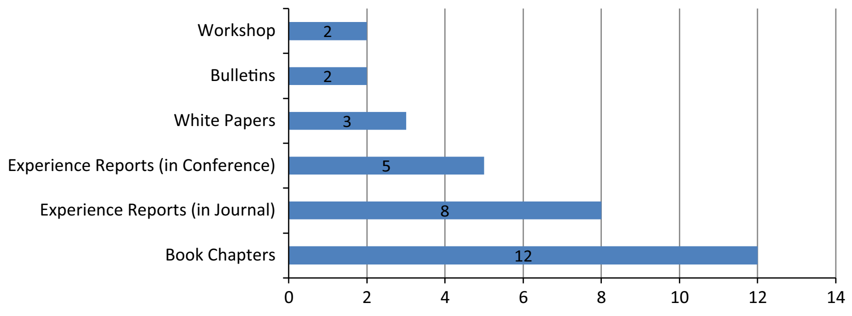

Fig. 6. Research type: Opinion Papers.

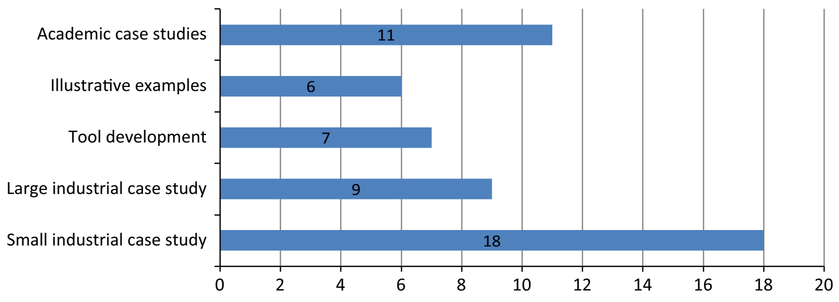

Fig. 7. Evaluation type: Solution Proposals.

these results. We also found that many studies introduce new ideas and the solutions to the problems regarding use case specification research, but fail to establish their contributions properly and show their validity in larger contexts. We argue that the use of evidence-based techniques for evaluation of the future studies may help in establishing more scientific and repeatable results.

4.3. RQ1: Evolution of the use cases

Use cases were first conceptualized by Jacobson in 1987 to capture the functional requirements of a software system in a plain text paragraph style format [78]. Jacobson's proposed use cases were intentionally informal to enable users to apply them when documenting specifications.

Use cases are written in natural language so that the specified functionalities can be understood by the stakeholders. Hence, there may be a possibility that users with different backgrounds, contextual knowledge, and experience may read and interpret the use case specifications differently [32,151]. The significant and efficacious utilization of functional requirements in the use cases makes them a popular technique for the purpose of eliciting, documentation and validation [32,134]. But, unavailability of formal representation of some natural language may result in confusion, difficulties and varied opinions in understanding the user requirements [51,70,126,130]. Without a standard style and a set of guidelines, it is difficult for the use case developers to construct high quality use case specifications.

It has been a common practice to use a template for specifying the use case specifications because the predefined structure (i.e., according to the applicability contexts) of a use case template helps the developer to identify and include the elements in each use case [32,93]. The first use case template proposed by Jacobson et al. [74] includes various use case elements such as use case name, preconditions, postconditions, basic, sub and alternate flow fields to specify the user requirements. This template mainly focused on describing the basic and alternate flow of events. With its acceptance into the UML community, they have become widely accepted prevailing approach for specifying the functional requirement of a software system [114].

Thereafter, a significant amount of research has been conducted in order to (1) formalize the basic flows and the alternate flows (2) propose a set of rules and guidelines (e.g., CP [36], CREWS (Co-operative Requirements Engineering With Scenarios) [2], etc.) for specifying the flow of events. Several templates by Harwood [63], Schneider [127], Mattingly et al. [99], Jaaksi [77] and Toro et al. [143] have also been proposed in order to improve the textual specifications further. The main focus of these templates were to enhance the representation capabilities of basic and alternative flows of events.

Cockburn [32] aspired to capture more complete set of information, and gives guidance on use case specifications and how to write them. The author proposed to include fields such as scope, level of abstraction, trigger, and special requirements in the use case template. The rationale behind the structure was to focus on the intent with more detailed information about the main flow, alternate flow, variations and extension points. Referable to the increased usages of the use cases in different software development activities (e.g., requirement elicitation, requirement validation, analysis modeling, testing, and model transformation), further efforts have been made in order to increase the formalism in the use case template. The formalism in the use cases was incorporated by (1) breaking the flow of events into various fields (2) making use of keywords, natural language restrictions. The evolution of use cases from 1992 to February 2014 is shown in Fig. 8.

We found twenty use case templates proposed and applied in various software development life cycle activities. Fig. 9 shows the plot of twenty use case templates and their citation rates. Based on the literature, we observed that the use case templates share some common fields such as use case name, preconditions, actors, basic flows, alternative flows, postconditions as shown in Table 6, where 'Y' indicates the presence of a specific property. The initial column of the table denotes the list of use case elements obtained by compiling the use case templates whereas the other column indicates the references of the proposed twenty use case templates. The description of the existing twenty use case templates are enumerated in Table 7.

Insfrán et al. [71] proposed a three-column format (i.e., general, actor/system communication, and a system response) use case template to structure the flow of events in the use case specifications. The general column describes the general activities of the system that should not be implemented but may help in understanding. Another column, actor/system communication specify the actions performed by the actors when interacting with the system, and the last one represents reaction of the system w.r.to the action performed by the actors (i.e., change of state). The aim of this template is to generate the conceptual analysis models from the use case specifications and to provide a way to move from requirements to a conceptual schema in a traceable manner. The use of a three-column format helps to maintain traceability between the requirements and the conceptual models. The use case template proposed by Somé [131] consists of eight fields that made use of various keywords (e.g., AND, AFTER, INCLUDE, EXTEND, and DELAY) to specify the flow of events in a restricted form of textual specifications. The rationale behind this template was to generate domain analysis models and objects related information from the use case specifications. Liu et al. [98], relies on the Rational Unified Process (RUP) template proposed by Kruchten [92], to generate domain analysis models – the class diagrams from the use case specification. This approach encounters the use of some natural language processing to extract the domain objects related information.

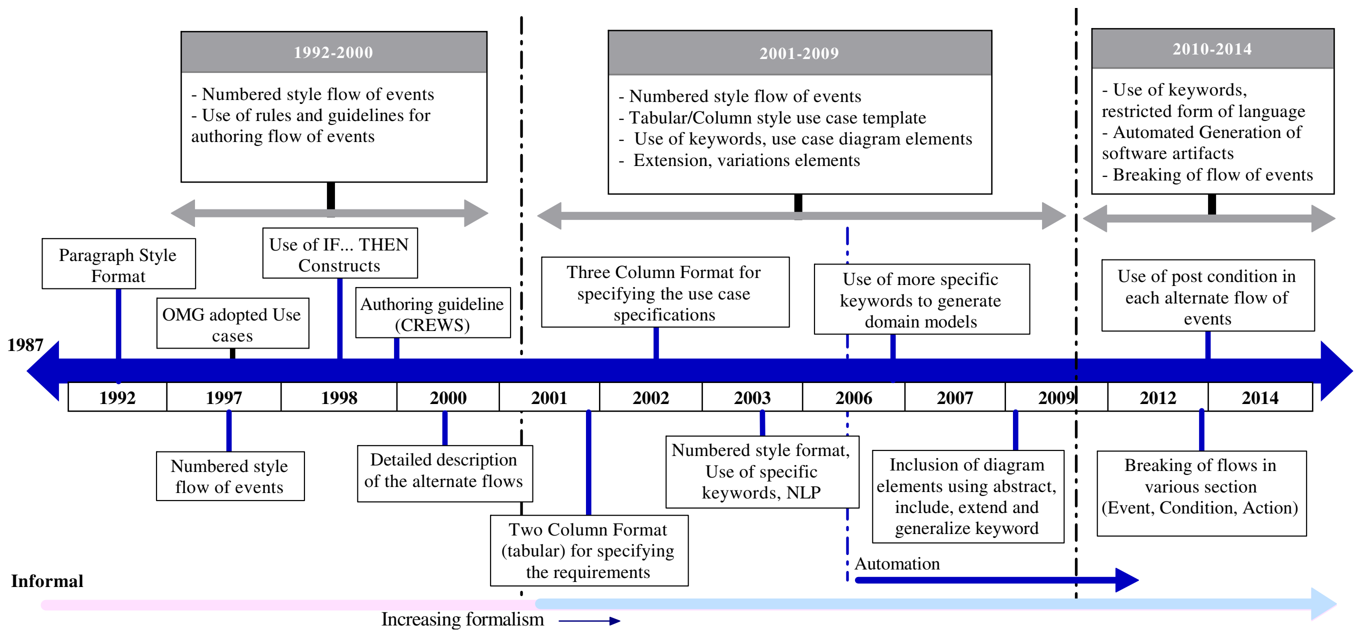
1992-2000
- Numbered style flow of events
- Use of rules and guidelines for authoring flow of events
2001-2009
- Numbered style flow of events
- Tabular/Column style use case template
- Use of keywords, use case diagram elements
- Extension, variations elements
2010-2014
- Use of keywords, restricted form of language
- Automated Generation of software artifacts
- Breaking of flow of events
Paragraph Style Format
OMG adopted Use cases
Use of IF... THEN Constructs
Authoring guideline (CREWS)
Three Column Format for specifying the use case specifications
Use of more specific keywords to generate domain models
Use of post condition in each alternate flow of events
1987
1992
1997
1998
2000
2001
2002
2003
2006
2007
2009
2012
2014
Numbered style flow of events
Detailed description of the alternate flows
Two Column Format (tabular) for specifying the requirements
Numbered style format, Use of specific keywords, NLP
Inclusion of diagram elements using abstract, include, extend and generalize keyword
Breaking of flows in various section (Event, Condition, Action)
Automation
Informal
Increasing formalism

Fig. 8. Evolution of the use cases.

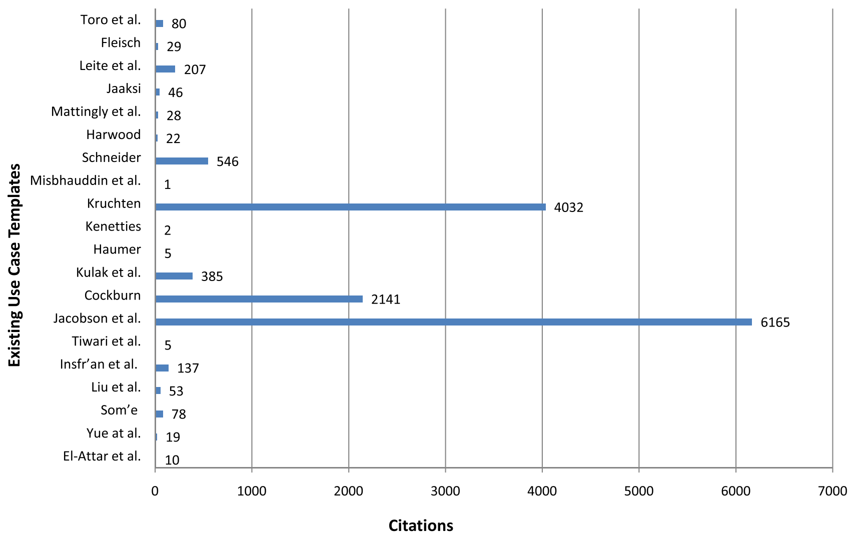

Fig. 9. Existing use case templates and their respective citations.

In order to capture the problem specifications in the use cases, software industries make use of Cockburn's use case template with some variations. For example, Kettenis's [86] use case template excluded the related information and variation sections mentioned in Cockburn's template. Moreover, they suggested to use the CP structure rules [36] to specify the dependent use cases in the specifications. The rationale was to develop detailed use case specifications with a careful writing convention. Haumer's [64] use case template makes use of various fields such as precondition, post-condition, basic and alternate flows, special requirements along with business rules to capture the user requirements. The ⟨*actor*⟩ stereotype is used to identify which actor is responsible for the execution of the event step in the use case specifications. The rationale of the template is to capture the requirements helpful in satisfying the business rules and hence is performance targeted.

More recently, there have been many efforts made to formalize use cases even further for the automated generation of software artifacts and, capturing a more complete, correct and consistent set of requirements. Yue et al. [151], proposed a use case template with alternate flows and basic flow with postconditions. The authors had classified the alternate flows into three types – specific, global, and bounded. The specific alternate flow refers to a specific step in the main usage scenario, the bounded alternate flow corresponds to several steps associated with the successful flow of events. And, the global alternate flow refers to any step in the flow. Some extra fields such as dependency with other use cases and generalization are also included in the template helpful in deriving the analysis models. Tiwari et al. [141], proposed a use case template where each specific flow is divided into the events, conditions and actions. The rationale behind the format is to categorize the events, conditions and actions, which is further used to validate the functional requirements. Misbhauddin et al. [102], extended the use case metamodel to facilitate the requirement analysis, verification and model transformation.

Table 6
Summary sheet: use case elements in various templates.

Use case element	[39]	[143]	[127]	[63]	[99]	[77]	[93]	[32]	[98]	[102]	[52]	[141]	[71]	[74]	[131]	[151]	[92]	[86]	[47]	[64]
Use case name	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Goal in context (Description)	Y						Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y		Y
Trigger							Y	Y									Y	Y		
Scope																		Y		
Summary			Y	Y	Y	Y	Y	Y												
Level of abstraction								Y										Y		
Stakeholders								Y										Y		
Stakeholders and their interests								Y										Y		
Primary actor		Y	Y	Y	Y	Y		Y		Y	Y	Y	Y		Y	Y	Y	Y	Y	Y
Secondary actors										Y	Y	Y	Y		Y	Y	Y		Y	Y
Special requirement								Y					Y				Y	Y		Y
Frequency								Y												
Priority								Y												
Performance target								Y												Y
Precondition	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y		Y	Y	Y	Y	Y		Y
Postcondition (Success)		Y	Y		Y	Y	Y	Y	Y		Y	Y		Y	Y	Y	Y	Y		Y
Postcondition (Failure)								Y										Y		
Due date		Y																		
Super use case								Y		Y										
Sub use case								Y		Y										
Dependency												Y				Y				
Generalization												Y				Y				
Main flow	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Sub flow	Y									Y		Y		Y						
Alternate flow	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y
Exception path	Y	Y																		
Exceptional flows								Y				Y								
Extension points			Y	Y	Y			Y		Y	Y	Y				Y	Y	Y	Y	
Variations								Y			Y	Y						Y		
Cross reference to high-level requirements													Y							
Open issue(s)								Y										Y		
Use of keywords								Y		Y		Y			Y	Y		Y	Y	Y
Constraints (conditions)										Y		Y								
Notes											Y									

Subsequently, a new use case template has been proposed to facilitate the constraint analysis and to model the use case elements information derived from the use case metamodel extension.

Based on the existing studies about the use case templates, we observed that the less-formal or loose nature of the use cases, on one hand provides flexibility to specify the requirements, but also open up multiple interpretations or ambiguities in the specifications. On the other hand, a more formal use case template can be more precise, but will require more amount of effort to specify the requirements. Also, the applicability of existing use case templates proposed in different contexts has not been reported yet. Each use case template have varying degree of formalism and a specific style of documenting the problem specification, which may affect its use in different software development processes [10].

We found as many as twenty use case templates through two-stage search and selection process. All these twenty use case templates were obtained from the research papers/technical reports (e.g., Insfrán et al. [71], Somé [131], Tiwari et al. [141], Yue et al. [151]), the industry white papers (e.g., Kettenis [86], Haumer [64]), experience reports (e.g., Mattingly et al. [99]), and the book chapters (e.g., Cockburn [32], Kruchten [92]). These templates represent considerable efforts and deliberations made by various researchers/practitioners during use case evolution. For instance, Somé [131] introduced control flow structures for iteration and concurrency; Yue et al. [151] suggested to make the use of the postcondition constraints after the completion of each alternate flow; Tiwari et al. [141] suggested to have the constraints for each flow of events in the use case textual descriptions.

Here, we may argue that these use case templates are not semantically equivalent as some template is better in some aspect but lacks on some other aspects. Specifically, Insfrán et al. [71] use case template captures the information about the adjacency pairs (i.e., actor to system communication) but not the dependency relations, step related constraints and the pre- post conditions. Cockburn [32] emphasizes on specifying more detailed information in a template in order to facilitate communication among the stakeholders, which may not be suitable for automated processing/translation of the textual descriptions. Somé [131] captures keyword-oriented information for the automated generation of domain analysis models from the use case specifications, which may reduce the communicability aspect of the use case specification.

4.3.1. Which use case template information and style is most useful in practice?

Though use case models have been significantly discussed and evolved in the existing literature, no such study has been conducted in the literature to judge the quality and applicability aspects of these use case templates. Having so many use case templates to document the textual descriptions raises many questions

Table 7
Use case templates: Context, style and description.

Context	Use case template	Style	Purpose/Description	Year
Requirement documentation	Jacobson et al. [74]	Paragraph style format	It was the first use case template proposed and used for documenting a problem specifications	1992
	Harwood [63]	Numbered style format, but does not include the post conditions	This describes how template can be used in other software activities	1997
	Jaaksi [77]	Numbered style format, use of sub flows	It was used for specifying the user specific functional requirements to improve the readability and understanding of the desired functionality	1998
	Mattingly et al. [99]	Numbered style format	It was used to generate the collaboration cases from the use case specifications	1998
	Schneider [127]	Numbered style format	This make use of specific constructs such as if…then and others to specify the basic and alternate flow of events	1998
	Toro et al. [143]	Numbered style format, included exceptional flows but not alternate flows	This describes a requirement elicitation approach to identify user requirements	1999
	Leite et al. [39]	Structured number style format	This describes how a particular scenario can be constructed in order to tackle important problems regarding scenario management	2000
	Cockburn [32]	Detailed, Tabular and Two column format	This describes how the detailed description of the use case specifications can be specified in a two-column tabular format	2001
	Haumer [64]	Numbered style format with paragraph style alternate flows	The inclusion of actor information in each flow of events such as ⟨*actor*⟩ to specify user requirements	2004
	Kruchten [92]	Numbered style formats	This makes use of keywords and in addition to standard template components has a specific section to specify non-functional requirements	2006
	Kettenis [86]	Numbered style formats, variations, extensions	It was used for specifying the user requirements in a linear format	2007
	El-Attar et al. [47]	Paragraph style format	This make use of some specific keywords such as abstract, include and extend to reducing inconsistencies in use case specifications	2009
	Kulak et al. [93]	Numbered and detailed flow of events	It was used for specifying user specific requirements, but the author's proposed rules do not permit the use of pseudocode and conditional/loop constructs while specifying the use case specifications in a template	2012
Domain modeling	Insfrán et al. [71]	Three column format	Conceptual modeling	2002
	Liu et al. [98]	Numbered style format, Use of specific keywords, NLP	Generation of class diagrams	2003
	Somé [131]	Numbered style formats, Keywords	Domain analysis models	2006
	Yue et al. [151]	Numbered style, specific use of postconditions and alternate flows	Analysis model generation	2013
	Misbhauddin et al. [102]	Numbered style format, sub flows and alternate flows	Use case metamodel facilitating the generation of analysis models, planning & estimation	2013
Requirement validation	Fleisch [52]	Numbered style, Two Column format	Requirement validation of component based software systems	1999
	Tiwari et al. [141]	Events, conditions, actions, sub flows and keywords	Validating use case requirements	2012

regarding the various usability aspects, such as user comprehension, applicability and effort required to learn.

For this, we analyze these 20 use case templates based on their research methodology stated in the paper. Fig. 10 shows the statistics for the use case templates where we found that no such evidences about their practical implications (i.e., no study falls under Evaluation Research Category). The five use case templates namely Insfrán et al. [71], Somé [131], Leite et al. [39], Yue et al. [151] and El-Attar et al. [47] were evaluated by conducting some small experiments/case studies; six use case templates namely, Liu et al. [98], Misbhauddin et al. [102], Fleisch [52], Tiwari et al. [141], Toro et al. [143], and Mattingly et al. [99] were proposed to provide the solutions to the problem, and rests were based on the experiences of the researchers gained from the industries or academia.

Among all, two [64,86] use case templates were commonly used in the industrial context, but their practical implications, i.e., understandability, learnability and operability, were not evaluated. Hence, it is an important issue that needs to be investigated in order to identify a such use case template whose information and style is more effective and efficient in terms of usefulness (e.g., increased understandability, efficiency in documentation), usability (e.g., time to learn) and their applicability in different contexts. It will also be interesting to investigate the right degree of formalism and the correct level of detail to be incorporated in the use cases for the developers to be better off.

To the best of our knowledge, there is only one empirical study that has been carried out in academic settings with relatively small number of subjects and small-scale projects for evaluating the usefulness of these use case templates [139]. But, for the same we found no such study carried out in industrial domain. Therefore, similar to this, more replicated studies can be performed in industrial settings with professional developers to evaluate the usefulness of the use case templates for a more practice-oriented outcome.

While scrutinizing these 20 use case templates, we found that they all have been used in three major contexts – specifying requirements, generating analysis models and validating requirements, following some useful structural and behavioral patterns. In the next section, we discuss and try to identify those patterns for each of three contexts.

4.3.2. Analysis of the use case templates

Our systematic review study helped us to identify twenty different use case templates proposed and applied in different contexts. We classified these twenty use case templates by their context of application, which we categorized under three major theme headings; requirement documentation, requirement validation and domain modeling. Table 7 lists the templates as per assigned category heading and gives a brief description about each use case template.

For each theme, we analyze the use case templates for their strengths and weakness in specifying the functional requirements.

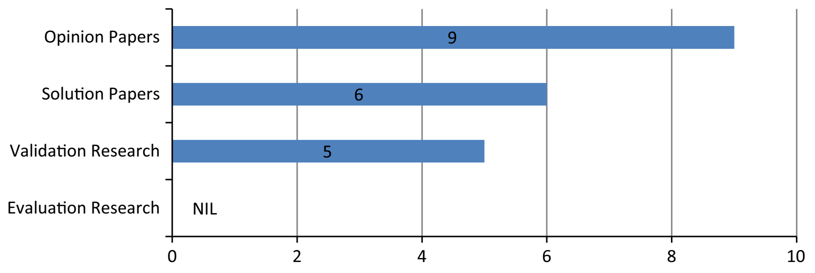

Fig. 10. Statistics for the use case templates: research types.

This resulted in the identification of patterns such as use case elements, style, and level of formality to specify the requirements in the use cases. Here, the problem specification of a Point-Of-Sale (POS) system [32] is used for illustration.

4.3.2.1. Requirement documentation. First, the thirteen use case templates classified under the requirement documentation theme were scrutinized to identify the commonalty and the differences. Several recommendations are also given to specify the requirements, shown below:

1. It is an important practice to specify the use cases such that they are comprehended by the users who are associated with the system [32,64,86].
2. To enable better understanding of the use case templates, information about the actor must be associated with each flow of events [64].
3. Comprehension of the templates can be improved by explicitly mentioning specific keywords and terminologies used in the template [32,92].
4. Developers were free to use any of the authoring techniques such as CP, CREWS, SSUCD's and others to describe the user requirements either by following a bullet points, paragraph or numbering style format [32].

We found that all these 13 use case templates emphasizes on specifying the main flows and the alternate flows. Each use case template makes use of a single-column format to specify the textual descriptions in a plain text. Table 8 shows the distinctive structure of the use case template that can be utilized to specify the use case textual descriptions, specifically for the requirement documentation purpose. This kind of structure typically help to support better communications (or understanding) between the stakeholders and the software developers [32,64,86].

4.3.2.2. Domain modeling. Next, we consider five use case templates that were classified under this theme. The proposed use case templates are intentionally formal, restricted only to use a numbered style format for specifying the requirements in the main flow and alternate flow sections. Some specific keywords such as VALIDATE, AFTER, DELAY, MEANWHILE, INCLUDE, EXTEND, IF. . .THEN, DO-UNTIL, and GO TO, have been used to facilitate the automated generation of domain analysis models and other software artifacts (e.g., natural language processing, generating quality analysis models such as UML Activity, Class, Sequence and the State diagrams). There have been several recommendations available for the same, some of them are listed as follows:

1. The minimal use of natural language constructs help to reduce the redundancy in the use cases, helpful in deriving quality analysis models [86,98,151].
2. "The terminology should be consistent throughout the use case specifications and should not impose unnecessary constraints on the design" [10].
3. Every alternate flow or exceptional flow must return to some specific step/flow of the use case specification [74,140,151].

Additionally, we also observed that the consistent use of terminology, keywords and the use case elements are useful for the same purpose. Likewise, these observations can also be applied for the planning and estimation purpose, where the details of the main actors and the prioritized functionalities must be seized. Also, the use cases applied for the same purpose must consist of information about the included and extended use cases in the specifications, along with their significant level of details while specifying requirements in the use case template [10,8]. Table 9 shows the distinctive structure of the use case template that can be used to specify the use case textual descriptions for the purposes of modeling and designing, planning and estimation, and automated text processing.

4.3.2.3. Requirement validation. Finally, the remaining two use case templates classified under the third category used for the requirement validation purpose makes use of a two-column format. These use case templates require some amount of processing of the textual descriptions to validate the quality of specified use case textual descriptions. Therefore, they admit a greater degree of formalism for specifying the problem specifications in various sections of the use case templates. This sort of structure forces the developers to specify the relationship between the use cases and their resemblance with the flow of events. This also includes the information about the pre and postconditions, actors and their associated flow of events.

Furthermore, it was likewise noted that the use case specifications used for generating the test requirements emphasizes on the completeness and correctness of the pre- post conditions and the relations between the actors and the use cases [10]. The purpose of a two-column format specifying the actor–system response format is more efficacious for bringing forth the test cases from the specifications [27,54]. Table 10 shows the distinctive structure of the use case template that can be used for requirement validation and testing purposes.

Table 11 shows the presence or absence of a specific property in a use case template, according to their applicability contexts. The overall assessment of these twenty use case templates highlighted their utilization and applicability in different contexts and purposes. Although, it is not clear which template is most efficient in a specific context (e.g., capturing requirements, domain modeling, testing, automated analysis, etc.). Hence, it will be interesting to investigate the applicability and usefulness (i.e., strengths and weaknesses) of each use case template.

4.4. RQ2: Applicability of the use cases

In this section, we present an overview of the selected studies applicable in different software development life cycle activities. There have been a lot of researches conducted, not only to specify the functional requirements in the form of textual descriptions, but

Table 8
Use case structure: requirement documentation.

Use Case Name: Process Sale
Actor: Cashier
Precondition: Customer arrives at a POS checkout with the items to purchase
Postcondition: Sale is saved. Receipt is printed. Stock data updated. Payment authorization approvals are recorded
Main Flows: *Single column paragraph, bullet or numbering style format*

1. Cashier starts a new sale
2. Cashier enters item identifier

. . .

5. System calculates and presents total price
6. Customer pays and System handles payment

. . .

9. Customer leaves with receipt and goods.

OR

(A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.)

Alternate Flows:
2a. Invalid identifier: System signals error and rejects entry
2b. There are multiple of same item: Cashier can enter item category identifier and the quantity

Extension Point:
5a. Process Gift Coupon Sale UC
. . ...

Variations:
6a. Paying by cash: (UC Handle Cash Payment)
6b. Paying by credit: (UC Handle Credit Payment)
. . .

Table 9
Use case structure: domain modeling.

Use Case Name: Process Sale
Actor: Cashier
Precondition: Customer arrives at POS checkout with goods to purchase.
Main Flows: *Single column/Two-column Format*
Use of Keywords such as VALIDATE, AFTER, DELAY, MEANWHILE, INCLUDE, IF. . .THEN, DO-UNTIL, GO TO, RESUME etc, and Looping constructs

1. Cashier starts a new sale
2. Cashier enters item identifier.

. . .

6. Customer pays amount and the System handles payment
INCLUDE Handle cash payment, INCLUDE Handle credit payment

. . .

9. Customer leaves with receipt and goods.

Postcondition: Sale is saved. Receipt is printed. Stock data updated. Payment authorization approvals are recorded.
Alternate Flows:
Use of Keywords such as If. . .then, RFS, EXTEND

6b. Customer says they intended to pay by cash but do not have enough cash MEANWHILE Customer uses an alternate payment method.
7a. If the items stock gets below a predefined minimum place a reposition order THEN GO TO Step 1.
. . .

Extension Point:
5a. Process Gift Coupon Sale UC
. . ...

Variations:
6a. Paying by cash: (UC Handle Cash Payment)
6b. Paying by credit: (UC Handle Credit Payment)
. . ..

Table 10
Use case structure: requirement validation.

Use Case Name: Process Sale
Actor: Cashier
Precondition: Customer arrives at a POS checkout with the items to purchase
Postcondition: Sale is saved. Receipt is printed. Stock data updated. Payment authorization approvals are recorded
Main Flows: *Use of Keywords such as INCLUDE*

Actor [Events]	**System Response [Actions]**
1. Cashier starts a new sale	POS System starts a new session
2. Cashier enters item identifier	POS System retrieve item information from the catalog system
3. Catalog system records sale line item, item description, price, and running total	POS system calculates and presents total price
4. Cashier tells the total, asks for payment	Customer pays the amount
. . .	
8. POS System prints receipt	Include UC Print Receipt

Alternate flows:
Use of Keywords such as IF. . .then, RFS, EXTEND
3–4a. IF Customer is eligible for the discount THEN Cashier enters the coupon ID and use Process Gift Coupon Sale UC ELSE Customer pays
. . .

Extension point:
3a. Process Gift Coupon Sale UC
. . ...

Variations:
4a. Paying by cash: (UC Handle Cash Payment)
4b. Paying by credit: (UC Handle Credit Payment)
. . ..

Table 11
Presence or absence of a property.

S. no.	Property	(I)	(II)	(III)
1	Level of formality *(Low, Medium, High)*	Low	High	High
2	Precondition	√	√	√
3	Postcondition	√	√	√
4	Information about secondary actors	√	√	√
5	Special Requirements/Open Issues	√	–	–
6	Information about included/extended use case	√	√	√
7	Postcondition for each alternate flow	–	√	–
8	Conditions for each flow of events to happen	–	–	√
9	Adjacency Pairs	–	–	√
10	Control flow structures for iteration	–	√	√
11	Information about Concurrent transactions	–	√	√
12	Flow of Events (Main, alternate and exceptional)	√	√	√
13	Extension Point and Variation	√	√	√

also how to use them in various software development activities such as requirement documentation, requirement analysis, requirement validation, domain modeling & implementation, testing, planning and estimation, and, maintenance. To answer the second research question (RQ2), we divide the selected studies into seven distinct categories according to their use within a precise software development process as shown in Table 12. Table 13 presents the category and papers in which the use cases have been applied where, it can be pointed out that there has been a less number of studies conducted in the *maintenance* activity.

Although, use cases have been largely applied on various software development activities but a lot of research has been conducted in order to improve the quality of use case specifications for their use in different software activities. In the following sections, we discuss the studies with strong impact on the use cases and an overview of the categories.

4.4.1. Requirement documentation

Established on the studies highlighting the use of the use cases for the documentation purpose clearly depicts that the use cases

Table 12
Categories for scrutinizing the selected studies.

Category	Aspects
Requirement documentation	Requirement elicitation, constructing use case specifications, writing efficient use cases
Requirement analysis	Dependencies, conflict detection, goal identification, priorities determination
Requirement validation	Inspection of the use case specifications for the purpose of identifying the issues
Domain modeling	Generation of analysis models and other software artifacts
Test case generation	Automated test case generation, model based testing
Planning and estimation	Effort Estimation, cost estimation, management tools
Maintenance	Change impact analysis (how efficiently the developers change the use case specifications?)

have been acknowledged as a popular approach for specifying the functional requirements [2,79,48,6]. Nevertheless, the major weakness with the use case specifications are particularly in terms of the documentation style and the pattern [10,3,114]. Several efforts have been made by various researchers to improve the quality of use case specification by making the use of rules, guidelines and the semantic structures [2,32,122,93,23]. The rules or guidelines proposed by the authors for writing the use case specifications help the user to specify the requirements, by allowing certain type of grammatical constructions. For instance, Cox [37], proposed the CR rules for authoring the use case specifications, allowing to make the use of subject-verb-object to frame the sentence structures. El-Attar et al. [47] presented a new structure named Simple Structured Use Case Description (SSUCD), to construct the quality use case specifications. Yue et al. [151], presented and validated the 26 set of rules for deriving the analysis models from the use case specifications. However, they still have disagreements in their use in different contexts, as the rules/guidelines used for one specific purpose may not be applicable in other purposes. Some of them are listed as follows:

1. How the flow of events in the use cases should be described, in a paragraph style, a numbered style format or in a column (actor-to-system response pairs) format style? [32,93,67].
2. For handling the alternatives, Schneider et al. [127] and Achour et al. [2] recommended to use if…then construction, while Kulak et al. [93] disagree to use such constructs in the use case specifications to increase the readability and understandability.
3. Achour et al. [2] recommended to use pseudocode in each flow of events description, while Kulak et al. [93] claimed not to use pseudocode, as this thins out the readability and comprehensibility of the specification.
4. Bittner et al. [23], suggested not to use ≪include≫ and ≪extend≫ relationships to assemble functionally decomposed use cases, whereas Cockburn [32], Yue et al. [151], Tiwari et al. [141], El-Attar et al. [47] and others, recommended to use them while specifying the flow of events in the use cases.
5. Do not include non-functional requirements (Bittner et al. [23], Jacobson et al. [74], Overgaard et al. [108]), but Kruchten [92] and Cockburn [32] considered them as a part of the use cases and recommended to specify them outside the description of the flow of events, as a separate section named ‘special requirements’.

Documenting the functional requirements of a software system in the use cases require, a precise and clear set of guidelines that helps the developer to specify them. The low comprehension and the differences in interpretation of the use case specifications may lead to a low quality software product [15,10,11]. The different set of guidelines may be applicable in different software activities, consequently it is important to distinguish the set of guidelines according to their applicability contexts. In this context, several studies have been proposed highlighting the use of rules, checklists and some set of empirical assessments/evaluations to improve the quality of use case specifications. In the following sections, RQ3 discusses these aspects of the use case specifications in detail.

4.4.2. Requirement analysis

Another major phase where the use cases have been applied and used is the requirement analysis. The studies in this category focuses onto the identification of useful information from the use case specifications in the early phases of the software development processes (e.g., goals, system information, defects, dependencies, requirement conflicts, etc). Belgamo et al. [21] presented a content based reading technique named Technique for Use Case Model Construction and Construction-based Requirements Document Analysis (TUCCA). This technique includes the identifications of the actors, their goals and the system’s use cases. An experiment was also conducted to compare the effectiveness of using TUCCA against the checklist or perspective based reading technique. The results of the study suggested that the use of TUCCA helps the users to detect defects in the requirement document as compared to the checklist or perspective based reading techniques.

Misbhauddin et al. [102] proposed an extended use case metamodel to facilitate model analysis and interchange by including the use case specification elements. The main objective of this extension is to provide a complete metamodel for the use case diagrams highlighting the elements and their relationships. The authors have also discussed the utilization of the proposed metamodel in various development activities such as model analysis, test case generation, effort estimation, use case metrics evaluation, and model transformation among modeling tools.

Table 13
List of all studies comes under different categories.

Focus	Papers	Count
Requirement documentation	[151,40,58,131,64,2,16,11,57,97,118,122,12,21,35,48,36,74,32,94,93,86,92,127,63,99,77, 112,6,45,39,143,108,67,107,132,145,126,137,100,144,23,19,79,37,146,101,61,3]	49
Requirement analysis	[5,21,73,124,46,66,141,102,123,83,96,65,62,17,51,130,120]	17
Requirement validation	[15,22,100,33,58,129,141,138,82,25,52,101,116]	13
Domain modeling	[149,136,56,135,87,13,119,151,53,121,131,14,60,98,71,148,94,128]	18
Test case generation	[104,27,106,54]	04
Planning and estimation	[103,8,105,80,7,26,29]	07
Maintenance	[18,10]	02

Hausmann et al. [65] proposed an approach to detect the conflicts among the functional requirements specified in the use cases. Here, the graph based transformation technique is used to capture the dependencies between the functional requirements expressed by different use cases. Thereafter, the so obtained graphs were then analyzed to detect the dependencies or conflicts between the functional requirements. The issues of interference between the different use cases were also mentioned, but does not investigated them further. Kanyaru et al. [83] presented an approach to analyze the dependencies among events within the use case specifications. The approach augment the dependencies by adding pre and post conditions on the events. Finally, it was concluded that consideration of dependencies helps to identify hidden issues regarding the problem domain.

Various researchers have tried to propose the paradigm for application of the safety techniques to the software systems from the requirements. Fault Tree Analysis (FTA) initially used in mechanical engineering safety–critical applications, it has also been advocated to be applicable to the software engineering domain as Software Fault Tree Analysis (SFTA) to find vulnerable failure modes and errors [62]. Tiwari et al. [141] presented a formal approach to analyze and validate the functional requirements specified in the use cases using Software Fault Tree Analysis (SFTA) and to identify the failure modes with Software Failure Modes Effect and Analysis (SFMEA). The approach first map the requirement into success tree and fault tree, and then they were synthesized to obtain the cut sets for the identification of missing or incorrectly documented requirements. Thereafter, the combination of SFTA and SFMEA is used to identify the vulnerable failure modes of a software system. Helmer et al. [66] presented a case study to analyze the intrusion using SFTA in order to determine the requirements for an Intrusion detection system.

Another important use of the use cases to elicit requirements from the users point of view was presented by Lee et al. [96]. Lee et al. [96] described a goal-oriented approach to structure and elicit requirements in the use cases, and to analyze and evaluate relationships between the requirements. The three step approach to construct the use cases are as follows:

1. Identify the primary and participating actors by investigating all the users associated with the system, directly or indirectly.
2. Identify goals based on a faceted classification scheme.
3. Build use-case models.

4.4.3. Requirement validation

With around 13 studies, it was found that there have been several inspection techniques available in the literature to improve the quality of use case specifications. The most used inspection technique for use case requirement validation is the checklist.

After eliciting and documenting the requirements in the use cases, it has been a common practice to use checklist based inspections to improve quality of the use case specifications [25,15,33]. Several recommendations from a number of checklist-based techniques has been presented in (e.g., [101,93,127]) order to formulate their own checklist applicable for a specific context. Wiegers et al. [146], presented a checklist to validate the use case specifications. The proposed checklist consider ambiguity of the use case textual descriptions as a quality attribute to improve the completeness and the abstraction of the specifications. Cox et al. [33], proposed a checklist to evaluate the quality of use case specifications. An experimental study was also conducted in order to illustrate the effectiveness of checklist in improving the quality of specifications. The results of the study suggested that the use of checklist help the group of students to identify the specification and internal defects. Whereas, in case of requirement defects (i.e., completeness and correctness of the specifications), no significant difference was found between the groups. Anda et al. [15], presented a taxonomy of typical defects and also proposed a checklist for detecting such defects that may exist in the use case specifications. Two experimental studies were conducted using undergraduate students as subjects to evaluate the applicability and effectiveness of the checklist. The results of the study suggested that the checklist-based approach are more useful in detecting the use case specification defects in comparison with the ad hoc based approach.

Kanyaru et al. [82], described a state-based approach to address the inadequacy of use cases for identifying the explicit dependencies that may exit in the use case specifications. The aim of this proposed approach is to provide an enhanced structure of the use cases to validate the requirement specifications for identifying the dependencies and interactions among them. Sinha et al. [129] presented an approach to validate the use case specifications by constructing an Abstract Syntax Tree (AST) from the problem specifications. The generated AST was then used for the automated inspection of the use cases. The contribution of this work is mainly emphasizes on to, (1) transforming the informal use case specifications into AST and (2) generating an integrated development environment, Text2Test for validating the specifications.

4.4.4. Domain modeling

After requirement documentation, with 18 studies, it was considered as the most valuable area for facilitating the automated derivation of domain models from the use case specifications.

The major challenges associated with the software development task are the problem of handling complexity, and the problem of managing changes. Arguably, models can play vital roles in meeting out those challenges. The development process typically consists of a series of model transformation activities to map the requirements and the executable code. Several studies have been conducted in order to generate domain analysis models or other software artifacts from the use case specification. Insfran et al. [71] proposed an approach to represent the requirements in a conceptual schema, i.e., in the from of use case specifications by means of a requirement analysis process. The main contribution is to introduce a specification model for capturing the functional requirements in terms of a conceptual schema to generate quality analysis models. This approach also includes a three-column use case template for facilitating better traceability between requirements and conceptual schema.

Somé [131] proposed an approach to support the use case based requirement elicitation, documentation, validation and simulation. This make use of a restricted form of natural language to generate the domain analysis model and the state model from the use case specification. And, the same has also been demonstrated using a tool support named Use Case Editor (UCEd). Subramaniam et al. [135], developed a tool supported approach – Use Case Driven Development Assistant (UCDA) for generating class diagrams. This tool provides assistance in developing use case diagrams, writing use case textual descriptions and deriving the analysis class diagrams from them. Ratcliffe et al. [119] investigated the application of use case specifications in the design phases of the software development process. At last, it was concluded that state based use case structures not only represent a succinct analysis format, but may also be used to map analysis models directly in the design process.

Yue et al. [151], proposed a restricted use case modeling approach (RUCM) to document use case specification in a specific format by applying 26 restriction rules to facilitate automated processing of the specification to generate class and sequence diagrams. The specific contributions of the study are (1) proposed an approach to automatically generate activity diagrams from the use case specification [149] and (2) an approach to facilitate the automated generation of analysis domain models from the use case specifications. Another approach of generating class diagrams from the use case specification was proposed by Liu et al. [98] that include two intermediate models (e.g., lexical analysis and the domain analysis) for the transformation. Kimour et al. [87], presented a systematic approach to identify objects from the use case model for the real-time embedded systems. The approach first transforms the use case specification into activity diagram, and second, the identification of the objects from the generated activity diagram, which were then used to derive domain analysis models. In continuation, we found several other semi-formal approaches that have been proposed (e.g., [53,60,121,148]) in order to derive analysis models from the use case specification. As the process involves some human intervention, thus may affect the completeness, the correctness and the redundancy of analysis models [151]. This suggests a need to have an automated approach for translating the use case specification into other static, behavioral and architectural models.

There have been several empirical studies have been carried out in order to evaluate the usefulness of the use case specification for deriving quality domain analysis models. For instance, Syversen et al. [136], presented the results of an empirical study to evaluate how a use case model can be applied in an object-oriented software development process. Twenty-six subjects were used to compare a use case driven process against a responsibility-driven process. The results of the study suggested that the use case driven process produced more realistic class diagrams with large variation in the number of classes. Moreover, a similar study with improved settings was reported by [13], where the results of the study suggested that the use case driven process generates better quality structured class diagrams.

We found that the generation of analysis models from the use case specification make use of some structured form of textual descriptions. However, the plausibility structure of a use case template that produces quality analysis models are still need to be investigated. Moreover, several empirical studies have been conducted to investigate the effectiveness of use case models for constructing domain analysis models [128,14], but were subjected to little empirical validation in both industrial and academic settings.

4.4.5. Test case generation

As we know that the freedom of specifying requirements in the use cases may arises some issues in the software development process. Therefore, to develop a quality software product, we need derive a clear and precise information from the use case specifications [104]. Testing a software system, consists of generating the test cases to determine and verify the behavior of the software products satisfying the requirements of the users and the stakeholders. Several studies have been conducted to generate the test cases from the use case requirements, but we have chosen those studies that make use of problem requirements to generate the use case specifications and then subsequently use them to generate the test requirements. Typically, the test cases were generated from the use case specifications by making use of various model driven approaches, where an intermediate model is derived followed by the derivation of the test requirements.

Fröhlich et al. [54], described an approach to generate the system-level test cases by following Cockburn's template [32] to specify the requirements. Each independent use cases are then transformed into a Statechart having the nested states and the transitions with actions and conditions, wherever applicable. Here, the sub and super states are generated by using the pre and post conditions of the use case specifications. The limitation concerns with the partial automation for generating the state diagrams from the use case textual descriptions. Another vital contribution for generating test cases from use case specifications can be found in [27]. Briand et al. [27] proposed an approach to generate the system-level test cases from the use case specifications. This approach elaborates the use cases with an activity diagram for each actor separately. The activity diagram was then transformed into a weighted graph to derive the regular expressions. Next, another set of regular expressions was generated by transforming use cases into sequence diagram. Finally, for each use case, their scenarios were replaced by the corresponding regular expressions to generate the executable test cases.

Use cases are the first good source of software testing [27,104] and have been successfully used to generate the test cases for further verification and validation of a software system. Despite this, the lack of formal representation of the desired functionalities specified in the use cases may affect the overall quality of the generated test requirements. Arguably, the same have also been highlighted in the literature, and suggested to use the use case template which include a precise definition about the pre- post conditions (i.e., for generating system level test cases) and the relation between the actors and the use cases [104,10].

4.4.6. Planning and estimation

According to our findings, we found that the published studies on planning and estimation is limited. Estimating the size of the software project in the early phase of the development is critically important for the planning and estimation purpose. Traditionally, the software size is estimated through the Function Points (FP), Number of Lines of Code (NLOC) and Object Points (OP), which are used as input to the COCOMO models [24]. But, due to the variations and problems in computing the FP, OP and NLOC, various UML based estimation techniques have been proposed.

The use case based effort estimation technique, Use Case Points (UCPs) has been introduced by Karner in 1993 [84]. The prerequisite of the method is to identify the use case specifications under development at a suitable level of detail [8,80,103]. Anda et al. [8], reported the results of three industrial case studies for estimating the effort based on the UCPs. The results of the studies supported the existing claims that use cases can be successfully used in effort estimation and also supports the expert knowledge in the estimation process. Mohagheghi et al. [103], modified the use case point method for the incremental large scale software development. They have modified the generated used estimation criteria for handling the use cases, associated actors, non-functional requirements and the team factors. Specifically, the extended method computes the effort and size of the very large projects over the multiple releases. Carroll et al. [29] presented a study to describe how the large organization can use the use case points to estimate the project cost and effort accurately. This study was conducted in the industry settings across hundreds of sizable projects which suggest that the accuracy of estimating project size deviates by 9% from the actual estimation cost of the project.

Nevertheless, the UCP technique has some limitations that (1) it does not consider internal structure of the use cases (2) may reduce accuracy of the estimation process [7,26]. For instance, use case has only 3 categories of complexity, i.e., simple, average, complex, independent of its internal structure. Suppose there are three use cases: UC1, UC2 and UC3 with 81, 4, 30 transactions respectively. The UCP method considers UC1 and UC3 as complex and UC2 as simple. But, it can be clearly seen that UC1 have a large number of transactions and is more complex than UC3. Therefore, a different method is required, which evaluate use case transactions separately. To overcome the aforementioned limitations, Barz et al. [26] introduced metrics based on the use case model named Use Case Size Points (USP). USP computes the project size by looking at the interior structure of the use cases whereby the number of actors, precondition, scenarios and their weight was estimated.

Our findings suggest that the use case can be successfully used in planning and estimation processes. However, the applicability and structure/design of the use case specifications may affect the overall estimation process [80,8]. Therefore, it is important to derive a use case structure for specifying the requirements in the use cases to have a precise effort estimation that would match with the expert's estimation.

4.4.7. Maintenance

According to our findings, we found that the published studies on software maintenance is very limited. No major study providing methods or tools on how to manage the use case specifications for the changes, release planning has been found in this study. An empirical study evaluating the impact of documentation on the software maintenance is presented by Arisholm et al. [18]. The results of their study suggested that the requirement documentation written without following some specific rules/guidelines was difficult to maintain in comparison with the formalized or standardized ones. Anda et al. [10], reported the findings of an experimental study conducted to assess the impact of making changes in the use case specification, specifically for a large software project during analysis modeling. The results suggested to focus on actors, pre- post conditions, descriptions of the basic flow, consistent abstraction level and the terminologies while making changes in the use case specification. As part of their conclusion they argue that systematic reviews on use case models are cost-effective. To achieve a more complete body of knowledge about the use case specifications related to their quality and the applicability, more empirical and systematic review studies need to be carried out in hereafter [151,140,48,93,115].

Evidently, we found lot of studies that have been practiced and applied in both industrial and research contexts for the purpose of documenting problem requirements in the use cases. Specifically, we found industrial relevance only in the requirement documentation, and the planning and estimation activity, whereas for other software development activities (see Table 12), we found numerous solution proposals and the empirical assessments highlighting the research relevance of the use case specification technique.

4.5. RQ3: Techniques used to assess the quality of specifications

Generally, four major techniques namely, the guidelines for authoring the specifications, checklist-based validation, quality assessment attributes and the empirical evaluations have been keyed out from the selected studies, used to evaluate the quality of specifications documented in the use cases are shown in Fig. 11. Fig. 12 shows the plot of techniques used to improve the quality of specifications from 1992 to 2014, whereas the selected studies lying under different techniques are shown in Table 14.

There have been numerous recommendations available in the literature suggesting what parameters constitutes the quality in use case specifications [15,12,2,19,32,33,93,114,47,45]. El-Attar et al. [45] described a set of 26 anti-patterns to improve the quality of use case specification. These anti-patterns "describes a commonly occurring solution to a problem that generates decidedly negative consequences [28]". These suggested anti-patterns are mostly used for capturing the functionalities in the use case diagrams, only fewer anti-patterns are suggested for describing the use case textual descriptions.

The overall quality of the use case specifications can be judged by (1) applying the inspection techniques (2) evaluating the quality attributes (3) conducting the experimental evaluations. We closely observed the selected papers to identify the quality attributes, specifically from the empirical evaluations, surveys, reviews that have been conducted to improve the quality of the use case specifications. Table 15 shows the quality attributes mentioned in the literature.

Typically, the defects in the use case specifications were identified by applying the inspection and review techniques. Many quality assessment validation techniques have been proposed in the literature to validate the use case specifications [33,11,15,47,25,10]. The checklist based approach for validating the use case specification was first introduced by Wiegers [146]. Wiegers's considers ambiguity of the use case descriptions as a criteria to improve the completeness and abstraction of the use cases. This validates the conformity of the use cases with the problem specifications and also determine whether the correct set of requirements have been drawn.

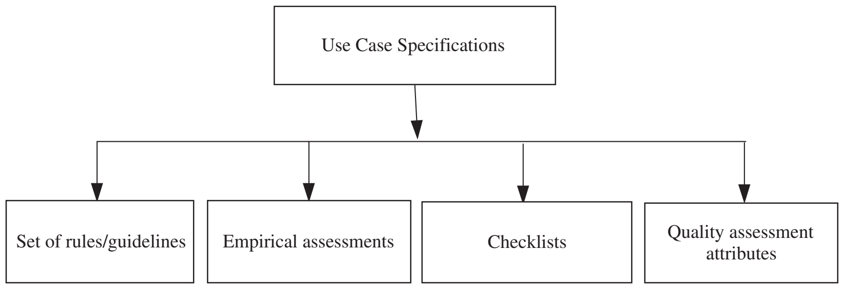

Fig. 11. Techniques used to judge the quality of specifications.

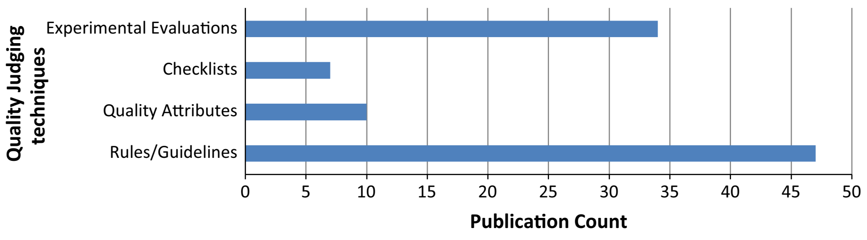

Fig. 12. Publication count for each technique.

Table 14
List of all studies comes under different techniques.

Taxonomy	Papers	Count
Rules/guidelines	[4,151,40,58,131,2,16,11,57,97,118,122,12,48,36,74,32,94,93,86,92,127,63,99,77, 112–114,6,45,39,143,108,67,107,126,44,35,31,23,19,79,37,146,101,61,3]	47
Quality assessment attributes	[10,144,47,139,146,114,12,11,18,37]	10
Checklist-based validation	[15,116,33,138,25,101,146]	07
Empirical assessment	[4,136,142,31,22,20,55,116,138,58,21,10,47,81,114,115,33,2,50,13,35,139,36,112,140,128,151,37,14,15,101,7,12,128]	34

Table 15
Quality attributes.

Quality attribute	Definition
Completeness [10,47,48,139,146]	There should be no missing functionalities about the – actors – pre and post conditions – flow of events (main flows, alternative flows, exceptional flows, sub flows) – relationship between the use cases and actors
Correctness [10,47,48,139]	There should not be – incorrect descriptions about the actors and their relationships – incorrect flow of events, incorrect alternative flows, pre and post-conditions
Consistency [10,47,48]	There should be no – descriptions of actors that are inconsistent with their behavior of the use cases – the structure of the use cases and the use of language and grammar should be consistent across all use cases – inconsistent use of keywords – inconsistencies among the use case descriptions, i.e., flow of events – inconsistencies in the use of alternative flows or numbering of events, or – pre- and post-conditions that are inconsistent with the use case descriptions
Understandability [114,12]	– the specification must be presented in a readable form – the information contained in the use case descriptions must be precise and unambiguous – the specification should also not contain any repeated information as this may lead to confusion
Ambiguity [10,139,146]	There should not be – actors that do not reflect their role – ambiguous descriptions of pre- or post-conditions – ambiguous descriptions of flow of events – ambiguous alternative flow conditions, or – any use of adverbs, adjectives, pronouns, synonyms, references, or negatives – ambiguous descriptions of the variations and extensions The ease with which the specifications can be read and understood. For this, the use cases should be precise and use a common terminology to specify the requirements
Level of abstraction [10,48,47]	The use case specifications should be written in a consistent level of abstraction. The GUI, design and implementation related details must not be included in the text
Maintainability [18,10]	The ease with which the changes can be made completely and consistently throughout the use case specifications without changing the semantic structure of the use cases
Verifiability [10,37]	The ease with which the specifications can be affirmed for their quality and defect free statements. This may include the use of natural language processing, checklist based approach and scrutinizing of each use case element
Communicability 7C's [114]	*Coverage* The use case should contain all that is required to answer the problem and extra unnecessary information specified is out of problem scope and is not required *Cogent* - Text Order: The use case should follow a logical path – Dependencies: The use case should complete as an end-to-end transaction (which can include alternative/exceptional flows) – The logic of the use case description should provide a plausible answer to the problem *Consistent Abstraction* – The use case should be at a consistent level of abstraction throughout *Consistent Structure* – Variations: Alternative and exceptional events should be excluded from the main flow and should be in a separate section – Sequence: Numbering of events in the main flow should be consistent *Consistent Grammar* – Simple present tense should be used – Adverbs, adjectives, pronouns, synonyms and negatives should be avoided *Consideration of the alternatives* – Viable: Alternatives and exceptions should make sense and should be complete – Numbering: Alternative and exception numberings should match the numbers in the main flow
Response time	– Time required to identify the information from the use cases – Time required to develop the complete, correct and consistent use case specifications

Anda et al. [15], presented a four point inspection checklist for evaluating the quality of use case specifications. This include the information about the actors, use cases, flow of events and relation between the use cases. Cox et al. [33,37], proposed a refined checklist which include the comprehensive details about the language, scope, alternatives, flow of events, abstraction, structure etc. Phalp et al. [115], presented an experimental study which evaluates the effectiveness of CREWS and CP guidelines for authoring the use case specifications. Specifically, these use case specifications were assessed using a use case quality description checklist. The experimental findings suggested that the CREWS set of guidelines performs better in terms of their ability to produce quality use case specification. The contents and details of the checklist items proposed in the existing literature are shown in Table 16.

To assist in writing the use case specifications using some natural language, several rules and guidelines have been proposed, particularly in terms of admissible structures [6,19,23], and based on the practitioner's experiences, several guidelines (e.g., CREWS [2], CP [36], CR [37] and SSUCD's [47]) are recommended to reduce ambiguities and inconsistencies [32,122]. For example, Cockburn [32] recommended to use '≪subject≫... ≪verb≫... ≪direct object≫... ≪prepositional phrase≫' to capture the flow of events in the use cases. The precise and correct use of set of rules/guidelines for specifying the use case specifications help the user/developer to specify the user requirements in the use case template. Several empirical researches have also been conducted to evaluate the effectiveness of these proposed guidelines [15,10,45,11].

The guidelines for constructing the use case specifications can be divided into three major categories as shown below. The examples of guidelines for each of the above mentioned categories are outlined in Table 17.

1. Guidelines that restrict the user to use natural language to some extent (constrain grammatical sentence structures, such as allowing only to use certain type of sentence format/structure such as subject-verb-object)
   - Language.
   - Structure.
   - Don'ts about the use case models.
2. Guidelines to specify the control structures and conditional statements.
3. Guidelines to specify the keywords.

Table 16
Use case validation checklist.

Context	Definition	
Scope of the Use case [33,32]	The use case should contain detail only relevant to the goal of use case – Is there any extra unnecessary information which is out of problem scope and is not required? – Does the description of each actor is consistent? – Does the use case textual descriptions is consistent with the goal of use case?	
Dependencies [74,134,65,32,83]	– Does the actor reach a state that stops the transaction from terminating as expected? – Do the use case diagram and the textual descriptions match? – Has the include-relation been used to factor out common behavior? – Does the behavior of a use case conflict with the behavior of other use cases?	
Abstraction [15,33,140,146]	The use case should be at a consistent level of abstraction throughout. Mixing abstraction levels such as problem domain, interface specification, internal design will cause difficulty in understanding – Is abstraction consistent?	
Structure [93,2,15,127]	Variations: Alternative and exceptional events should be excluded from the main flow and should be in a separate section. Inclusion of alternative paths in the main flow reduces readability. – Are alternative paths excluded from the main flow? – Are all the recoverable failures recorded as extensions? – Sequence: Numbering of events in the main flow should be consistent. Are there any inconsistencies?	
Language [151,140,15,33,2,32,61]	– Simple present tense should be used throughout – Adverbs, adjectives, pronouns, synonyms and negatives should be avoided. Have they been used? – Does active voice is used rather than passive voice?	
Consideration of alternative flows [33,32,93,2,15,74]	– Viable: Alternatives and exceptions should make sense and should be complete. Are they? – Numbering: Alternative and exception numberings should match the numbers in the main flow. Do they or is there inconsistency? – Separation: is there a separate section for alternative/exceptional paths to the main flow? – Is the numbering of flow of events in the main flow consistent?	
Goal of the use case [74,32,67]	– Is it phrased as a goal that succeeds? – Is the flow associated with main flows are clear and correct? – Is expected input and output correctly defined in each use case? – Does each event in the normal flow of events relate to the goal of its use case?	
Scenarios [74,32,15,33]	*Actors*	– Are there any actors that are not defined in the use cases? – Does the use case template consist of the associated actors (primary, secondary)? – Is there any actor- to-actor interaction in the descriptions? – Is it clear which actors are involved in which use cases, and can this be clearly seen from the use case diagram and textual descriptions? – Are all the actors connected to the right use case?
	Preconditions	– Are the pre- and post-conditions correctly described for all use cases, that is, are they described with the correct level of detail? – Do the pre- and post conditions match for each of the use cases and are they testable?
	Postconditions	Are all stakeholders' interests satisfied?
	Flow of Events	– Does the flow of events in the use case textual descriptions numbered correctly? – Does the use case documentation have a text or a flow describing the interaction between the actor and the system? – Does the use case documentation have possible alternative and exception flows? – Does the process move distinctly forward after successful completion of the step?

Table 17
Guidelines for authoring the use case descriptions.

Category		Guidelines
Natural language constraints [151,23,93,6]	Language [2,32]	1. Use Present tense [36] 2. Consistent Words 3. Simple Sentences 4. Active voice rather than passive voice. 5. The object of the use case can be only be a noun (example, inventory) or a qualified noun (in-stock inventory) [67] 6. Use declarative sentence only ("Is the system idle?" is a non-declarative sentence). 7. Use the form Subject verb object direct object-prepositional phrase (if any) [32,2]
	Structure [10,33,32]	1. The subject of a sentence in basic and alternative flows should be the system or an actor. 2. Describe the flow of events sequentially. 3. For naming use case either start with actor or list the use case followed by its initiating actor (Format: Action Verb + Qualified Object) [67]. 4. Describe one action per sentence and avoid compound predicates. 5. Each sentence in the description should be on a new, numbered line. Alternatives and exceptions should be described in a section below the main description and the sentence numbers should agree [36]. 6. There should be logical coherence throughout the description [36]. 7. When an action occurs there should be a meaningful response to that action [36]. 8. The use case name should consists of the keyword such as abstract, specialized, implements to show the relationships between the use cases [131]
	Don'ts [93,23,151,32,74]	1. GUI elements descriptions are not allowed in the use cases 2. Non-functional requirements must not be described in the use cases 3. Actor-to-actor interactions are not allowed 4. Avoid Compound Predicates 5. Avoid negatives [36] 6. Do not use modal verbs (e.g., might, could, should) 7. Avoid adverbs (such as very, more, rather, and the like) [36] 8. Do not use negatively adverb or adjective (such as hardly and never) 9. Avoid pronouns if there is more than one actor [36]
Control structures and conditional statements [127,92]		1. To specify conditional logic sentences (IF–THEN–ELSE) is used while specifying the use case textual descriptions [2] 2. 'Loop' ⟨repetition condition⟩ 'do' ⟨action⟩ [2] 3. For Resume to Step, Go To and Reference Flow (RFS) associated fields are used in the basic flow, alternate flows and sub flows. [140,151]
Keywords		1. INCLUDE, EXTEND keywords to describe use case dependencies include and extend [47,131] 2. INCLUDE Keyword is used in the Basic flow or Sub flow section while EXTEND keyword is used in the Alternate flow section [141] 3. ⟨action 1⟩ 'meanwhile' ⟨action 2⟩ [2] 4. Underline other use case names [86,36] 5. VALIDATE, AFTER, DELAY [151,131]

Derivation of use cases from the problem specification requires the set of rules and guidelines for the documentation; and the inspection techniques for the validation. Several empirical studies have been conducted in order to evaluate the impact of guidelines and checklists on the quality of use case specifications. Achour et al. [2], conducted two experimental studies to examine the impact of CREWS guidelines with regard to the completeness of the use case specification. Sixty-nine subjects were divided into four groups to develop the use case specification using CREWS content and style guidelines. The experimental results suggested that the (1) use of CREWS writing guidelines improves the quality of use case specification, (2) the impact of each set of guidelines is different, and (3) use cases may not be written correctly at once, thus can be corrected using some systematic techniques such as checklist or usage based verification.

Phalp et al. [114,115] conducted an empirical study to compare two sets of writing rules: the CREWS rules and CP rules [36]. The evaluation was made on the basis of seven quality factors, referred as the '7Cs of communicability' [112]. The experimental results indicated that the use of CP rules for authoring the specifications require less learning overhead than the CREWS guidelines, and also CP performs well, same as the CREWS. Yue et al. [151], proposed the set of 26 guidelines and a use case template, where sixteen guidelines are used to constrain the natural language issues and other 11 guidelines are used to make the use of keywords, and control structures. The intent of these guidelines is to restrict the users to use natural language in order to reduce the ambiguities in the specifications. This later facilitates the automated generation of the quality analysis models from the use case specifications. El-Attar et al. [47], presented the results of a controlled experiment conducted to evaluate the applicability of SSUCD using various quality assessment attributes. The results of the study suggested that the usage of SSUCD while deriving the use case specifications improves their consistency among the use case models.

Anda et al. [12] conducted an experiment to compare three different sets of guidelines: minimal guidelines (identification of actors and the use cases), template guidelines (documenting the problem specifications in a format), style guidelines (a modified version of the CREWS rules, focused on documenting the flow of events of each use case). The developed use case specifications were evaluated in terms of their quality and understandability. The results showed that the use of template guidelines derives highest quality and understandability of the specifications. Whereas, the style guidelines perform better than the minimal guidelines in terms of the information contents. It was also suggested that combination of style and template guidelines helps to improve the quality of use case specification. Cox et al. [35] conducted a replicated study to assess the impact of CREWS guidelines while authoring use case textual descriptions, and found no clear evidences about their significant impact on the quality of use case specification.

Porter et al. [116] conducted an experiment to compare the defect detection inspection methods, namely Ad-hoc, Checklist-based reading and perspective-based reading. The results of the study highlighted that the perspective based reading (the scenario approach) has more defect detection rate than the ad hoc and checklist based approach. Whereas, Miller et al. [101] replicated the same and found the similar results that the scenario approach is better than the checklist based approach. Another study of comparing the inspection methods was undertaken by Thelin et al. [138] to compare usage-based and checklist-based reading technique. The results suggested that users who applied the usage-based reading technique to identify the defects were found to be more efficient and effective than the users using the checklist-based reading technique.

Several other empirical studies have also been conducted to evaluate the applicability of the use cases in order to (1) evaluate the role of use case specifications in the construction of class diagrams [14], (2) compare different defect detection approaches used to evaluate the quality of the use case specifications [21], (3) investigate the applicability of different analysis model generation approaches based on the use cases [148], (4) automatically generate the domain models from the use case specifications using natural language processing [98], and (5) assess the impact of textual descriptions specified in a given use case template on the quality of generated domain analysis models [151].

Though, there have been a lot of researches conducted to formalize the use case specifications by making use of some set of guidelines and the checklists based approaches. But, due to the variation in the use of use cases and their style of the specifying specifications, each guideline or use case elements have a different impact. Based on the review study, total twenty use case templates have been identified that have been applied in different context and purposes, and hold different kind of semantic structure, where, nevertheless, different set of guidelines are applicable.

4.6. RQ4: Open issues and future directions

Based on our findings and observations for the aforementioned research questions, the design of this research question (RQ4) aimed to identify the unresolved issues that still need to be addressed to improve the effectiveness of use case specifications. There have been numerous studies conducted in order to (1) improve the quality of use case specifications (2) automate the software development processes (3) incorporate more amount of formalism facilitating their use in various software development activities. Though, a vast knowledge about the use case specifications is available in the literature but still there are some issues and the future aspects that need to be looked into further.

The rigorous research on the use case specification starts after they have been accepted as a specification documentation tool. A batch of variations of the proposed rules, guidelines and the templates were illustrated in the existing literature, but providing a complete, unambiguous, correct, consistent and expressible set of requirements for constructing high quality use case specifications, defect free statements are even questionable. Still, there are problems with the use case models due to their nature of specifying the requirements. If these problems are not resolved then it may cause design and implementation difficulties later in the development process.

Having so many variations of the use cases and their associated attributes such as the rules, the guidelines, and the quality assessment techniques raise many questions regarding their applicability contexts, degree of formalism, usability aspects, maintainability, traceability and quality. The identified issues those need to be addressed are enumerated as follows:

1. How to estimate the correctness and completeness of the use case specification derived from the problem specification? [12,47,48,50].
2. Identification of the use case elements that helps in deriving the domain analysis models from the problem specifications [53,98,128,151].
3. Estimation of the usability of various use case templates in different software development activities such as requirement documentation, requirement analysis, requirement validation, test case generation, planning and estimation, and maintenance [8,10].
4. How much formalism is enough? [32,93,139,140].
5. How to incorporate the efficient changes in the use case specifications to keep them updated with the change requirements in the specifications? [18].
6. How to fix the correct abstraction level for use case specification to avoid describing problem requirements at different levels?
7. Among twenty of them, which use case template information and style is most useful for practice in terms of their understandability, learnability and operability (according to the usability aspects, ISO/IEC 9126 [72,91])?

Apart from these issues, more experimental studies need to be conducted in the future to improve the quality of use case specification research. Though, we found that there have been several experimental studies conducted to evaluate the usability of use case templates in terms of their use in specifying, validating and analyzing the requirements both in industry as well as academic settings. But, more rigorous controlled experiments and empirical evaluations need to be carried out in order to establish some standards for assessing the effectiveness, efficacy and usability of the use case specifications.

Use cases have been largely used and applied to document the specifications in a template format, but the use of natural language while specifying the requirements retain some questions such as 'what level of formality can be reasonably attained by means of natural language?'. We observed that the colloquial or loose nature of the use cases provides flexibility to specify the requirements, but introduces the problems of multiple interpretations, ambiguities, misconceptions and inconsistencies in the specification. However, a more structured use case template may help in specifying the requirements more precisely, but will require extra effort and knowledge about the semantics/notations. The lack of a semi to automated tool support approach for specifying the requirements mean that the process is influenced by human ability, and may affect the quality and comprehensibility of the use case specification. Hence, a use of a semi to automated tool support for specifying the use case specifications may help the developers to manage the requirements accordingly.

Since use cases have been used, applied and evolved over a period of time from an initial plain text format to a more formal, structured format, the automated analysis and generation of various software artifacts is increasing. Thus, it is worth to simulate the applicability of the use case specifications in developing other software artifacts during the software development process. Additionally, the inter-operability of the use case specifications in different stages of the software development life cycle activities is even questionable.

5. Research findings

Use cases are acknowledged as an important requirement documentation technique for specifying the functional requirements of a software system. Several efforts have been made to formalize the use case specifications in order to optimize their utility in various software life cycle activities. The use case template used for documenting the specifications developed for one project may not be effectively used for other projects due to their individual specifications [32], and also, their applicability may vary accordingly [10]. Improved awareness of the contextual application of various use case templates may allow developers to produce better documentation by choosing more suitable templates for their projects. Therefore, the impact of use cases in various software development life cycle activities cannot be over-emphasized.

The use case specifications were typically employed in two different perspectives – one, for documenting the functional requirements and the other, for generating the lower-level software artifacts. With reference to a purely informative perspective the use case specification were documented in a tabular or a paragraph style format with minimal degree of formalism (e.g., [32,39,64,74,86,143], etc.). The rationale behind this was to elicit, analyze and validate the problem specifications with the viewpoint of stakeholders and developers. Referable to the increased usages of the use cases in different software development activities, the use cases have also been used for the purpose of generating lower-level software artifacts through a model-transformation process. For this, the degree of formalism and the level of detail in the use cases have been increased by applying the set of rules, keywords, natural language constructs, control structures and the conditional statements (e.g., [71,131,149,151], etc.). Arguably, the increased formalism in the use case specification helps in generating quality software artifacts, but, may also affect the usefulness (e.g., increased understandability, efficiency in documentation) and usability (e.g. time to learn) aspects [140,151].

From the literature reviewed, we summarize our findings against each research question that are as follows:

RQ1 *Use cases have evolved from paragraph format textual descriptions to a more formal keyword-oriented format for facilitating automated information retrieval*. We found as many as twenty use case templates that have been proposed and applied in different contexts. The use of a semi to automated tool supported approach may help the use case developers/business analysts to a certain extent to specify the quality requirements in the use cases.

RQ2 *Use cases have been applied and used in almost all the software development life cycle activities*: Use cases have largely been applied in a requirement documentation phase to specify the functional requirements of a software system. However, there still have some discrepancies with the proposed rules, guidelines, and grammatical constructs (also suggested by [10,48,93]). This is imputable to the pertinency of the use cases in different software development activities and the researchers have been proposed them in isolation without concerning the software activity/purpose for which the use case specifications to be used. Hence, their applicability, use of specific set of rules, guidelines and grammatical constructions should be targeted for specific purposes. We found less number of studies proposed for incorporating efficient set of changes in the use case specifications, thus need to be further investigated.

RQ3 *There have been lot of advices available to judge the quality of the use case specifications based on the checklists, guidelines, quality measures and empirical assessments but this needs further addressing*. A deeper understanding of the use case specifications and their corresponding measures to assess the desired quality is likely to improve the overall quality of the software product. There exist several works on improving the quality of use case specifications by applying the use case writing guidelines and the checklists, but there have not been studies on whether such techniques practiced in the industry.

RQ4 The issues that need to be further investigated are (1) the right degree of formalism (2) incorporating efficient changes in the specifications (3) the usability of various use case templates (4) estimation of completeness and correctness of the use case specifications (5) their applicability in developing various other software artifacts during the software development process (6) assessing the effectiveness of the quality assessment techniques for improving the quality of use case specifications (7) evaluating industrial or practical relevance of the use cases in different contexts.

The problem with the current research scenario in the use case specification research is a gap between the evaluation contexts – academic or industrial. The studies applied in academic contexts possess students as subjects and small-scale projects for assessing the effectiveness of a proposed technique/approach. These kind of studies lacks on their strength of the evaluation, hence, the results cannot be generalized as intended/identified. Our analysis revealed very few studies that were applied in industrial contexts, thus no industrial relevance can be claimed, and need to be analyzed in the future to evaluate the usefulness of the use case specification.

The recent trends of the use case specification research is moving towards the automated analysis of the software specification problem written in natural language for capturing the functional requirements of a software system, and the automated extraction of various software artifacts (e.g., events, actions, objects, constraints) to support traceability.

6. Discussions

The systematic review process is a long drawn process where emphasis is placed to increase the reliability of the search, study selection, data extraction and research outcomes. Accordingly, researchers might be more concerned about the cost effectiveness of conducting such systematic reviews. It may be interesting to estimate the time spent for the efforts while conducting the

Table 18
Highlights for the researchers.

Research direction	Issues
Evidence-based assessments	Lack of quality of synthesis, small set of replicated studies, missing industrial relevance, lack of evidences for automation/tool support, applicability of the use case templates, applicability of various rules, guidelines and checklists, judging the correct level of abstraction for use case specification, and the right degree of formalism to be incorporated in the use cases, application of use cases for developing safety–critical systems, etc.
Automated processing of the use case specification	Correct usages of various use case elements, identification of objects and their relationships, disambiguation, completeness and correctness of the extracted information, automated generation of quality analysis models from the use case specification, etc.
Comparative analysis of the use case evolution techniques	Identifying an inspection technique which reveals maximum defects, judging the usability aspects of various use case templates for the developers to be better off, effectiveness of the rules/guidelines used for specifying problem requirements in the use cases, etc.

systematic review. In our case, approximately three months of time, considering 6–7 h per person day, was spent in the review process. Specifically, 7–10 days for searching; 25–30 days for study selection; 15–20 days for data extraction and 20–30 days for the data analysis. This also accounts the time spent when two different authors collaborate in order to include or exclude a study. This resulted into the identification of the open issues, problems and the future aspects for both the researches and the practitioners.

Table 19
Highlights for the practitioners.

Usage/application scenario	Approaches/techniques
Rules/guidelines	A consolidated list of the rules/guidelines is shown in Table 17
Checklist	A consolidated list of checklist items is shown in Table 16
Use case template	Cockburn [32] (for the purpose of documentation) and Yue et al. [151] (for the automated generation of software artifacts)
Planning and estimation	Use case point size [7,103]
Domain modeling	[14,121,151,71,131]
Maintenance	[18,10]
Safety–critical system	[62,66]
Test case generation	[27,104]
Documentation	[2,48,112,140]
Formalism	[4,39,140,126,131,40,57,23,6,146,127]
Empirical assessments	[114,112,20,35,13,101,7,55,18]

Our review findings suggest that the researchers on the use case specification are now focusing their efforts towards evidence-based assessments, automated translation of use case descriptions to other software artifacts in the development process, and comparative analysis among various use case specification techniques. For instance, Anda et al. [14] investigated the role of use cases in the construction of class diagrams; Sinha et al. [129] presented a tool support approach for automated analysis of the use case specification supported by an industrial case study; and Cox et al. [36] conducted an experiment to compare two sets of use case authoring guidelines in terms of their admissible structures. We present a summary of the research directions, and issues in Table 18.

Many studies on use cases highlighted evidences on their applicability in different contexts and purposes. For instance, use cases have been successfully applied in various software development activities such as planning and estimation, domain modeling, test case generation and so on, where, the claims are well supported by applying case studies, developing tools or conducting empirical assessments. A summary of use case usages and application scenarios which can be of some interest to practitioners working in this domain is presented in Table 19.

6.1. Threats to validity

The inaccurate extraction of data, selection of literature resources, and exclusion of papers were considered to be the major threats against this review study.

Table A.20
Results of quality scores of selected studies: Evaluation and Validation Research Papers.

ID	Study	QA1	QA2	QA3	QA4	QA5	QA6	QA7	QA8	QA9	QA10	QA11	QA12	QA13	Score
S1	[2]	1	1	1	0.5	1	0.5	1	1	1	1	1	1	0.5	11.5
S2	[4]	1	1	1	1	1	0.5	1	1	1	1	1	1	0.5	12
S3	[14]	1	1	1	1	1	0.5	1	1	1	1	1	1	0.5	12
S4	[18]	1	1	1	1	1	1	1	1	1	1	1	1	0.5	12.5
S5	[20]	1	1	1	1	1	1	1	1	1	1	1	1	1	13
S6	[39]	1	1	1	1	0.5	1	0.5	1	1	1	NA	1	0.5	10.5
S7	[51]	1	1	1	1	1	1	1	1	1	1	NA	0.5	0.5	11
S8	[101]	1	1	1	1	1	1	1	1	1	1	1	1	1	13
S9	[116]	1	1	1	1	1	0.5	1	1	1	1	1	1	0.5	12
S10	[138]	1	1	1	1	1	1	1	1	1	1	1	1	1	13
S11	[142]	1	1	1	1	1	0.5	1	1	1	0.5	0	1	0.5	10.5
S12	[7]	1	1	1	1	1	1	1	1	1	NA	NA	1	0.5	10.5
S13	[29]	1	1	1	1	1	1	1	1	1	1	1	NA	NA	11
S14	[103]	1	1	1	1	1	1	1	1	1	NA	NA	1	NA	10
S15	[114]	1	1	1	1	1	1	1	1	1	NA	NA	1	NA	10
S16	[10]	1	1	1	1	1	1	0.5	1	0.5	1	0	1	0.5	10.5
S17	[12]	1	1	1	1	1	0.5	1	1	1	1	0	1	0.5	11
S18	[15]	1	1	1	1	1	0.5	1	1	1	1	0	0.5	0.5	10.5
S19	[21]	1	1	1	1	1	1	1	1	1	1	1	1	0.5	12.5
S20	[22]	1	1	1	1	0.5	0.5	1	1	1	1	0	1	0.5	10.5
S21	[33]	1	1	1	1	1	1	1	1	1	1	1	1	0.5	12.5
S22	[35]	1	1	1	1	1	1	1	1	1	1	1	1	1	13
S23	[36]	1	1	1	1	1	1	1	1	0.5	1	0	0.5	0.5	10.5
S24	[37]	1	1	1	1	1	0.5	1	1	1	1	1	1	0.5	12
S25	[47]	1	1	1	1	1	1	1	1	1	1	0	1	1	12
S26	[50]	1	1	1	1	1	1	1	1	1	1	0	1	1	12
S27	[55]	1	1	1	1	1	1	1	1	1	1	0	0.5	0.5	11
S28	[81]	1	1	1	1	0.5	1	1	1	1	1	0	1	0.5	11
S29	[112]	1	1	1	1	1	1	1	1	1	1	1	1	0.5	12.5
S30	[115]	1	1	1	1	1	1	1	1	1	1	0	1	0.5	11.5
S31	[128]	1	1	1	1	1	1	1	1	1	1	0	1	0.5	11.5
S32	[139]	1	1	1	1	1	1	1	0.5	1	1	0	0.5	0.5	10.5
S33	[140]	1	1	1	1	1	1	1	1	1	1	0	1	1	12
S34	[151]	1	1	1	1	1	1	1	1	1	1	0	1	1	12
S35	[136]	1	1	1	1	1	1	1	1	1	0.5	0	1	0.5	11
S36	[13]	1	1	1	1	0.5	0.5	1	1	1	1	1	1	0.5	11.5

Table A.21
Results of quality scores of selected studies.

ID	Study	QA1	QA2	QA3	QA4	QA5	QA6	QA7	Score
S37	[25]	1	0.5	1	1	1	1	1	6.5
S38	[26]	1	0.5	1	1	1	1	0.5	6
S39	[27]	1	1	1	1	1	1	1	7
S40	[30]	1	1	1	1	1	1	1	7
S41	[44]	1	1	1	1	1	1	1	7
S42	[45]	1	1	1	1	0.5	0.5	0.5	5.5
S43	[46]	1	1	1	1	0.5	1	0.5	6
S44	[52]	1	0.5	0.5	0.5	0.5	1	0.5	4.5
S45	[53]	1	1	1	1	1	1	1	7
S46	[54]	1	0.5	1	1	1	1	0.5	6
S47	[62]	1	1	1	1	1	1	1	7
S48	[65]	1	1	1	1	1	1	1	7
S49	[66]	1	1	0.5	1	0.5	1	1	6
S50	[71]	1	0.5	1	1	1	1	1	6.5
S51	[78]	1	1	1	1	1	1	0	6
S52	[82]	1	1	1	1	1	1	1	7
S53	[83]	1	0.5	1	1	0.5	0.5	0.5	5
S54	[98]	1	0.5	1	1	0.5	0.5	1	5.5
S55	[100]	1	1	1	1	1	1	1	7
S56	[102]	1	0.5	1	0.5	1	1	1	6
S57	[104]	1	1	1	1	1	1	1	7
S58	[106]	1	0.5	0.5	1	0.5	1	0.5	5
S59	[107]	1	1	1	1	0.5	1	0.5	6
S60	[119]	1	1	1	1	1	1	1	7
S61	[123]	1	0.5	1	0.5	0.5	1	1	5.5
S62	[126]	1	1	1	1	1	0	0.5	5.5
S63	[130]	1	1	1	1	1	1	1	7
S64	[131]	1	1	1	1	1	1	1	7
S65	[135]	1	0	1	1	0	1	0.5	4.5
S66	[141]	1	1	1	1	0	0.5	0	4.5
S67	[143]	1	0.5	1	0.5	0.5	1	0	4.5
S68	[149]	1	1	1	1	0	0.5	0	4.5
S69	[73]	1	1	1	1	1	1	1	7
S70	[87]	1	1	1	1	1	1	1	7
S71	[121]	1	1	1	1	1	1	0.5	6.5
S72	[105]	1	1	1	1	0.5	1	1	6.5
S73	[144]	1	1	1	1	1	1	1	7
S74	[31]	1	0.5	1	1	0.5	1	1	6
S75	[40]	1	1	1	1	1	0.5	1	6.5
S76	[48]	1	1	1	1	1	1	1	7
S77	[58]	1	1	1	1	1	1	1	7
S78	[80]	1	1	1	1	0.5	0.5	0.5	5.5
S79	[118]	1	1	1	1	1	1	1	7
S80	[122]	1	1	1	1	1	1	0.5	6.5
S81	[129]	1	1	1	1	1	1	1	7
S82	[148]	1	1	1	1	0.5	1	0.5	6
S83	[137]	1	1	1	1	1	1	1	7
S84	[56]	1	0.5	1	1	0.5	0.5	0.5	5
S85	[132]	1	0.5	1	1	1	1	0.5	6
S86	[145]	1	1	1	1	1	1	1	7
S87	[124]	1	0.5	1	1	1	1	0.5	6

ID	Study	QA1	QA2	QA3	QA4	QA5	Score
S88	[3]	1	1	1	1	1	5
S89	[5]	1	1	1	1	1	5
S90	[6]	1	1	1	1	1	5
S91	[8]	1	1	1	1	1	5
S92	[11]	1	0.5	1	1	1	4.5
S93	[16]	1	0.5	1	1	1	4.5
S94	[17]	1	1	1	1	1	5
S95	[19]	1	1	1	1	1	5
S96	[23]	1	1	1	1	1	5
S97	[32]	1	1	1	1	1	5
S98	[57]	1	0.5	1	1	0.5	4
S99	[60]	1	1	1	1	1	5
S100	[61]	1	1	1	1	1	5
S101	[63]	1	0.5	1	1	1	4.5
S102	[64]	1	1	1	1	1	5
S103	[67]	1	0.5	1	1	1	4.5
S104	[74]	1	1	1	1	1	5
S105	[77]	1	1	0.5	1	1	4.5
S106	[79]	1	1	1	1	1	5
S107	[86]	1	0	1	1	1	4
S108	[92]	1	1	0.5	1	1	4.5
S109	[93]	1	0.5	1	1	1	4.5
S110	[94]	1	1	1	1	1	5
S111	[96]	1	1	1	1	1	5
S112	[97]	1	1	1	1	1	5
S113	[99]	1	1	1	1	0.5	4.5
S114	[108]	1	1	1	1	1	5
S115	[113]	1	1	0.5	1	1	4.5
S116	[120]	1	0.5	0.5	1	1	4
S117	[127]	1	1	1	1	1	5
S118	[146]	1	1	1	1	1	5
S119	[125]	1	1	1	1	1	5

6.1.1. Publication bias

We have considered all such studies that were published in different forums utilizing the use case specification technique for their proposals/evaluations. We believe that limiting the set of literature resources reduces the possibility of generalizing the review outcomes. This may introduce the risk of missing technologies, experiences, evidences and evaluations published in other forums such as technical reports, workshops, white papers, experience reports and book chapters. As the major publication venues (see Table C.23) are included in the systematic review by following the two-stage search and selection process, hence threat should be limited [75,76].

With respect to the selection of literature resources, we include six meta-search database engines to search the relevant studies about the use case specification research. These databases are very well-known for storing the significant amount of researches published in the reputable journals and conferences related to the computer science and software engineering. Arguably, if some publications were missed out from the search process due to limiting our search to six databases, then this would not introduce any bias or variations in the review outcomes. Apart from this, we manually search papers in the top ranked journals and the conferences, specific to the requirement engineering research domain. The references and citations list of the selected studies were also reviewed in order to get more studies that we might miss out and may be relevant. More importantly, we have not explicitly searched for the technical/experience reports, book chapters, bulletins, and white papers, hence identified by reviewing the references of the selected relevant studies.

We argue that most of the papers relevant to the use case specification research and their applications were covered in the review process. But, it might be few cases missed by our approach of search and selection. For instance, we additionally search the papers in the venues specific to the conceptual modeling, natural language processing, specification tools, elicitation, testing, and, model checking (e.g., CAiSE, ICST, ISSTA, ASE, STTT, etc.) and found approximately 10–15 additional studies. Though, these studies are relevant but could not substantiate the claim. We found that most of the papers make use of a use case specification technique without discussing its relevance or notations. This means that the search missed papers should not introduce any systematic bias with respect to the review results.

6.1.2. Searches

Six electronic databases were searched to identify the studies relevant to our review protocol. To reduce omission from this process, we also searched for the papers in potentially relevant, peer-reviewed journals and conferences related to the topic of the systematic review. The search query was applied to the titles and abstract and then expanded to include the papers from the reference and citation lists. The purpose of the two-stage search process (see Section 3.2.3) was to ensure that the vast majority of key papers have been included. The search process include general as well as specific terms [75,150] related to the use case specifications to identify the relevant papers.

The assessment of the search process was based on the number of unique papers identified through hand searching. The initial manual searching of relevant papers in seven journals and three conferences revealed a total of 51 studies, which were considered as the start-set articles that ought to be found while performing keyword-based search in six electronic databases. Before using the studies for data extraction and synthesis, we perform the validation of the search process by identifying the unique studies that were found through the keyword-based automated search process. Here, we found 46 unique studies, out of 51 relevant studies. This suggests that the effectiveness of the keyword-based automated

Table B.22
Example of the papers excluded from the systematic review process.

Authors name	Title of the study	Research type	Reason for exclusion
K. S. Wasson	A case study in systematic improvement of language for requirements	Solution Proposals	Title and abstract Scrutiny
E. Knauss, C. El Boustani	Assessing the quality of software requirements specification	Validation Research	Title and abstract Scrutiny
F. J. Lange Christian	Improving the quality of UML models in practice	Evaluation Research	Title and abstract Scrutiny
J. Cleland-Huang	Quality requirements and their role in successful products	Evaluation Research	Title and abstract Scrutiny
F. Chantree, B. Nuseibeh, A. De Roeck, A. Willis	Identifying nocuous ambiguities in natural language requirements	Solution Proposals	Title and abstract Scrutiny
C. Yunan, S. Il-Yeol	Guidelines for developing quality use case descriptions	Opinion Papers	General discussion, lack of evaluation (quality assessment exclusion)
E. B. Fernandez, J. C. Hawkins	Determining role rights from use cases	Solution Proposals	Full paper text exclusion, missing use case relevance/notations
D. Elenburg	Use cases: background, best practices and benefits	White paper	General opinions
I. Omoronyia, G. Sindre, M. Roper, J. Ferguson, M. Wood	Use case to source code traceability: the developer navigation view point	Solution Proposals	Full paper text exclusion
M. Riebisch, M. Hbner	Refinement and formalization of semi-formal use case descriptions	Opinion Papers	General discussions
N. Seyff, N. Maiden, K. Karlsen, J. Lockerbie, P. Grnbacher, F. Graf, C. Ncube	Exploring how to use scenarios to discover requirements	Evaluation Research	Full paper text exclusion, not use case oriented
S. Espana, N. Condori-Fernandez, A. Gonzlez, O. Pastors	An empirical comparative evaluation of requirements engineering methods	Validation Research	Full paper text exclusion, not use case oriented
I. Menzel, M. Mueller, A. Gross, J. Doerr	An experimental comparison regarding the completeness of functional requirements specifications	Validation Research	Full paper text exclusion, consists only comparative study, and not use case oriented
D. Michal, S. Peretz, S. Arnon	Comparing the impact of the OO-DFD and the use case methods for modeling functional requirements on comprehension and quality of models: a controlled experiment	Evaluation Research	Full paper text exclusion, missing use case relevance/notations, consists only comparative study
T. Yue, L. C. Briand, Y. Labiche	A use case modeling approach to facilitate the transition towards analysis models: concepts and empirical evaluation	Solution Proposals	Full paper text exclusion, more complete paper is included
K. Cox	Cognitive dimensions of use cases: feedback from a student questionnaire	Opinion Papers	Lack of evaluation (quality assessment exclusion)
K. Cox, R. Jeffery, A. Aurum	A use case description inspection experiment	Opinion Papers (experience report)	No new information can be depicted, general discussions
D. G. Firesmith	Use cases: the pros and cons	Opinion Papers (bulletins)	General opinions
H. Gomaa	Designing concurrent, distributed, and real-time applications with UML	Opinion Papers (book chapter)	Missing use case relevance/notations
R. R. Hurlbut	The three Rs of use case formalisms: realization, refinement, and reification	Opinion Papers (white paper)	Missing use case relevance/ notations, general opinions
S. Lauesen	Software requirements: styles and techniques	Opinion Papers (book Chapter)	Not use case oriented
K. Phalp, K. Cox	Guiding use case driven requirements elicitation and analysis	Opinion Papers	Missing use case relevance/ notations, lack of evaluation
O. Ryndia, P. Kritzinger	Improving requirements specification verification of use case models with SUSAN	Opinion Papers (technical report)	Full paper text exclusion, missing use case relevance/notations
J. Ryser, M. Glinz	A scenario-based approach to validating and testing software systems using Statecharts	Solution Proposals	Title and abstract scrutiny
D. Amyot, H. Xiangyang, H. Yong, D. Yong Cho	Generating scenarios from use case map specifications	Solution Proposals	Title and abstract scrutiny
A. Belgamo, S. Fabbri	Constructing use case model by using a systematic approach: description of a study	Validation Research	Full paper text exclusion, more complete paper is included
A. B. Nassif	Enhancing use case points estimation method using soft computing techniques	Solution Proposal	Lack of evaluation (quality assessment exclusion)
N. Bolloju, S. X. Sun	Exploiting the complementary relationship between use case models and activity diagrams for developing quality requirements specifications	Validation Research	Lack of evaluation (quality assessment exclusion)
M. El-Attar, J. Miller	Matching anti-patterns to improve the quality of use case models	Solution Proposals	Full paper text exclusion, more complete paper is included
S. McGee, D. Greer	Towards an understanding of the causes and effects of software requirements change: two case studies	Validation Research	Full Paper text exclusion, not use case oriented

search process and the query strings was 'substantial' (0.90). Specifically, we may argue that the use of keywords following the 'use case; use cases' as a prefix in the query string limit us to select a wide range of studies (i.e., for those studies which have applied use cases to show their proposed application relevance).

6.1.3. Studies included/excluded for the synthesis

The studies were selected on the basis of the search strategy which include (a) the existing literature electronic databases (b) selection criteria. Total 119 papers were selected using the two-stage search and selection process from all literature resources. To boot, a precise definition of the selection criteria complied with the research questions was enforced to avoid incorrect exclusion of the desired studies.

One of the major potential threat may be the subjective judgments made to include or exclude a study, and to classify papers in four different categories for the equality assessment. To include or exclude a paper from the review process, the initial scrutiny was done by reading the papers' title and abstract. This introduces a threat, as the title and abstract may not reflect what is actually presented in the paper. For this, we together read papers' full text and then agreed on reaching consensus. Additionally, to limit this threat, (1) the process used for inclusion or exclusion, and the classification schemes were tested prior to the systematic review, (2) at each step of the review process, the discussion was made among the authors to reach final consensus, and (3) the classification schemes and the quality assessment questions used in our study for the investigation were based on [147].

6.1.4. Data extraction

There has been a vast amount of research conducted on the use case specifications, and our study is concentrated on answering four research questions about their (1) evolution, (2) applicability, (3) quality assessments and (4) the issues/problems that needs to be investigated further. A data extraction form was used to assess the relevant data from the selected primary which help in

Table C.23
List of the selected studies considered for the systematic review.

Publication type	Source	Refs.
Journal	IEEE Transaction on Software Engineering	[138,18,62,104,116]
Journal	ACM Transaction on Software Engineering and Methodology	[151]
Journal	Information and Software Technology	[121,10,73,144,81,87,105,119,137,131,148]
Journal	Requirement Engineering Journal	[17,25,39,40,51,58,66,71,82,118,128]
Journal	Empirical Software Engineering	[14,35,47,101,55]
Journal	Journal of System and Software	[20]
Journal	Software Quality Journal	[114,115]
Journal	Software and System Modeling	[27,48,79,46,102]
Journal	Journal of Object Technology	[16]
Journal	Journal of Research and Practice in Information Technology	[22,33]
Journal	The Open Software Engineering Journal	[57]
Journal	Journal of Object-Oriented Programming	[63,77,99]
Journal	Data and Knowledge Engineering	[122]
Journal	Software development (Springer)	[146]
Journal	International Journal of Software Engineering and Knowledge Engineering	[132]
Journal	Computing and Informatics	[4]
Conference	IEEE Requirement Engineering Conference	[2,50,142,125,120,53]
Conference	Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications	[29,100,78]
Conference	International conference on Software Engineering	[65,103,11]
Conference	International conference on Model driven Engineering Language and Systems (UML)	[8,107,106]
Conference	International symposium on Empirical Software Engineering and Measurement	[21,13,7]
Conference	International Conference on Empirical Assessment in Software Engineering	[37,112]
Conference	European Conference on Object-Oriented Programming	[12,54]
Conference	Annual International Computer Software and Applications Conference	[26]
Conference	International Conference on Software Engineering and Knowledge Engineering	[15,135]
Conference	International Conference on Software Engineering Research, Management and Applications	[45]
Conference	International Conference on Technology of Object-Oriented Languages	[97]
Conference	IEEE International Symposium on Object-Oriented Real-Time Distributed Computing	[52]
Conference	IEEE Canadian Conference on Electrical and Computer Engineering	[98]
Conference	International Conference on Software Testing, Verification and Validation	[129]
Conference	IEEE International Conference on Dependable Systems Networks	[130]
Conference	Asia Pacific Software Engineering Conference	[139,124]
Conference	Indian Software Engineering Conference	[140]
Conference	IEEE International Conference on Engineering of Complex Computer Systems	[141]
Conference	Australasian computer science conference	[123]
Conference	International Conference on Business Information System	[31]
Conference	International Conference on Conceptual Modeling	[56]
Conference	Fundamental Approaches to Software Engineering	[145]
Conference	Annual Hawaii International Conference on System Sciences	[136]
Conference	International Conference on Recent Advances in Natural Language Processing	[30]
Conference	International Conference on Software Engineering Advances	[44]
Conference	European Conference on Modeling Foundations and Applications	[149]
Conference	International Conference on Software Engineering and Computer Systems	[80]
Workshop	International Workshop on Requirements Engineering: Foundation For Software Quality	[36,83,143]
Workshop	International Workshop on Use Case Modeling: Use Cases in Model-Driven Software Engineering	[60]
Workshop	International workshop on software process simulation modeling (in conjunction with ICSE)	[113]
Workshop	IEEE Software Engineering Workshop	[126]
Book Chapters	Addison-Wesley/ACM Press	[3,23,32,74,93,61,92,108,127]
Book Chapters	Pearson Education	[6,19,94]
Bulletins	IEEE Software	[5,96]
White papers	IBM, oracle	[64,67,86]

analyzing the data more rigourously. To reduce the inaccuracy of the data extraction and the reviewer biases towards a specific technique/approach/paper, the analysis was carried out independently by the authors using the quality assessment and the content assessment criteria. The decision to include or exclude a paper from the systematic review was taken by collating the information extracted by each author, and tried to be rigorous in assessing the reason for their disagreements, if any. The final data reported was agreed by both the authors in order to assess the study reliability and relevance.

As the authors came from academic domain, we believe that the possibility of favoring a specific use case specification technique has been ruled out. Since this exercise was conducted to (1) capture the knowledge about the use case specification research carried out so far, (2) identify the current research status, and (3) highlight the issues and future prospects that need to be investigated, in order to improve the quality of use case specifications and their application.

6.1.5. Analysis of the use case templates

Contempt of the threats to validity discussed above about the extraction of data, selection and rejection of the papers, there may also be a threat to validity for the selection and analysis of 20 identified use case templates.

We believe that it was hard to search for the use case templates from the studies because they have been advised to accomplish any particular intention of capturing the requirements for the software development. Hence, we manually searched these 119 papers patiently in order to identify the use case templates proposed in the respective subject areas. For instance, Misbhauddin et al. [102], proposed a new format for the capturing the use case elements stated in the proposed use case meta-model, and Insfrán et al. [71], proposed a new format for generating the conceptual models from the use case specification.

The analysis presented in this paper evaluates the effectiveness of these use case templates proposed and applied in different contexts. Therefore, the comparison is made on the basis of their contexts and the specified textual descriptions. The aim was to assess their effectiveness in terms of their structures and the elements they composed of. Although, we believe that this does not claim the strength or utility of a particular use case template, and the results highlighted the applicability of a structure in different contexts and the quality of textual descriptions specified in the use case templates.

7. Conclusions and future work

The challenges to correctly, consistently and completely specifying the user requirements using some linguistic factors are difficult, and acknowledged by various researchers. Several efforts have been made to communicate the requirements with greater fidelity among the stakeholders through some natural language restrictions and the language constructs. But, due to the varying degree of formalism in the use cases, it was difficult to set some standards. Use cases are primarily used in the requirement documentation phase due to their communicative nature, while in other phases more amount formalism is required for automating the subsequent processes.

The aim of this systematic literature view is to identify and examine the applicability of the use case specification technique in various software development activities. A total of four research questions was formulated in order to identify and ascertain existing use case templates, the studies related to specifying the problem requirements in the use cases, the techniques used to judge their quality, and the limitations of the use case specification research. This was carried out through identifying, assessing and interpreting 119 relevant studies. A total of twenty use case templates was compiled from our search. We found no such study that has been reported so far in order to identify a use case template whose information and style is most useful in practice, hence the empirical studies need to be carried out in both academic and industrial settings.

The primary studies identified in this review varied, i.e., most of the studies are carried out in academic contexts using small case studies or illustrative examples. This indicates that the evidences on use case usefulness could not be widely acceptable and substantiated further. Hence, there is a need to establish strong evidences in order to make the existing research outcomes useful in industrial contexts. Specifically, the issues that needs further addressing are (1) the identification of a use case template for each particular purpose and context where the use cases are applicable and effectively used (2) the efficient change management (3) the tool support approach for judging their quality (4) estimating completeness of the documented specifications (5) evaluating the applicability and usefulness of the use cases in industrial contexts.

Appendix A. Quality assessment

See Tables A.20 and A.21. The primary studies S1–S15 belongs to 'Evaluation Research'; S16–S36 to 'Validation Research'; S37–S87 to 'Solution Proposals'; and S88–S119 to 'Opinion Papers' category.

Appendix B. Example of excluded studies

Here, we show the examples of such studies that has been excluded from the review, after analyzing the paper's full text and after assessing their relevance using quality assessment questions (see Table B.22).

Appendix C. Publication venues and corresponding primary studies

Table C.23 shows the list of 119 studies selected for the systematic review of use case specification.

References

[1] P. Achimugu, A. Selamat, R. Ibrahim, M.N. Mahrin, A systematic literature review of software requirements prioritization research, Inf. Softw. Technol. 56 (6) (2014) 568–585. <http://www.sciencedirect.com/science/article/pii/S0950584914000354>.

[2] C.B. Achour, C. Rolland, C. Souveyet, N.A. Maiden, Guiding use case authoring: results of an empirical study, in: Proceedings of the 4th IEEE International Symposium on Requirements Engineering, RE '99, IEEE Computer Society, Washington, DC, USA, 1999, pp. 36–43. <http://dl.acm.org/citation.cfm?id=647646.731122>.

[3] S. Adolph, A. Cockburn, P. Bramble, Patterns for Effective Use Cases, Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] B. Alchimowicz, J. Jurkiewicz, M. Ochodek, J. Nawrocki, Building benchmarks for use cases, Comput. Inform. 29 (1) (2012) 27–44.

[5] I. Alexander, Misuse cases: use cases with hostile intent, Software, IEEE 20 (1) (2003) 58–66.

[6] I. Alexander, R. Stevens, Writing Better Requirements, Pearson Education, 2002.

[7] B. Anda, H. Benestad, S. Hove, A multiple-case study of software effort estimation based on use case points, in: 2005 International Symposium on Empirical Software Engineering, November 2005, pp. 41–49.

[8] B. Anda, H. Dreiem, D. Sjøberg, M. Jørgensen, Estimating software development effort based on use cases – experiences from industry, in: M. Gogolla, C. Kobryn (Eds.), UML 2001 The Unified Modeling Language. Modeling Languages, Concepts, and Tools, Lecture Notes in Computer Science, vol. 2185, Springer, Berlin Heidelberg, 2001, pp. 487–502. http://dx.doi.org/10.1007/3-540-45441-1_35.

[9] B. Anda, K. Hansen, I. Gullesen, H. Thorsen, Experiences from introducing UML-based development in a large safety-critical project, Emp. Softw. Eng. 11 (4) (2006) 555–581. http://dx.doi.org/10.1007/s10664-006-9020-6.

[10] B. Anda, K. Hansen, G. Sand, An investigation of use case quality in a large safety-critical software development project, Inf. Softw. Technol. 51 (12) (2009) 1699–1711. http://dx.doi.org/10.1016/j.infsof.2009.04.005.
[11] B. Anda, M. Jørgensen, Understanding use case models, in: Proceedings of Beg, Borrow and Steal Workshop, International Conference on Software Engineering, June 2000.
[12] B. Anda, D. Sjøberg, M. Jørgensen, Quality and understandability of use case models, in: J. Knudsen (Ed.), ECOOP 2001 Object-Oriented Programming, Lecture Notes in Computer Science, vol. 2072, Springer, Berlin Heidelberg, 2001, pp. 402–428. http://dx.doi.org/10.1007/3-540-45337-7_21.
[13] B. Anda, D. Sjøberg, Applying use cases to design versus validate class diagrams – a controlled experiment using a professional modelling tool, in: Proceedings of the 2003 International Symposium on Empirical Software Engineering, ISESE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 50–60. <http://dl.acm.org/citation.cfm?id=942801.943618>.
[14] B. Anda, D. Sjøberg, Investigating the role of use cases in the construction of class diagrams, Emp. Softw. Eng. 10 (3) (2005) 285–309. http://dx.doi.org/10.1007/s10664-005-1289-3.
[15] B. Anda, D. Sjøberg, Towards an inspection technique for use case models, in: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, SEKE '02, ACM, NY, USA, 2002, pp. 127–134. http://doi.acm.org/10.1145/568760.568785.
[16] B. Anderson, Formalism, technique and rigour in use case modelling, J. Object Technol. 4 (6) (2005) 15–28. <http://dblp.uni-trier.de/db/journals/jot/jot4.html#Anderson05>.
[17] A.I. Antón, R.A. Carter, A. Dagnino, J.H. Dempster, D.F. Siege, Deriving goals from a use-case based requirements specification, Require. Eng. 6 (1) (2001) 63–73. http://dx.doi.org/10.1007/PL00010356.
[18] E. Arisholm, L. Briand, S. Hove, y. Labiche, The impact of UML documentation on software maintenance: an experimental evaluation, IEEE Trans. Softw. Eng. 32 (6) (2006) 365–381.
[19] F. Armour, G. Miller, Advanced Use Case Modeling: Software Systems, Pearson Education, 2000.
[20] A. Ottensoosera, J.M. Feketea, C. Menictasd, Making sense of business process descriptions: an experimental comparison of graphical and textual notations, J. Syst. Softw. 85 (3) (2012) 596–606.
[21] A. Belgamo, S. Fabbri, J. Maldonado, TUCCA: improving the effectiveness of use case construction and requirement analysis, in: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering, ISESE '05, November 2005, IEEE Computer Society, pp. 266–275.
[22] B. Bernárdez, A. Durán, M. Genero, Empirical evaluation and review of a metrics-based approach for use case verification, J. Res. Pract. Inf. Technol. 36 (4) (2004) 247–258.
[23] K. Bittner, Use Case Modeling, Addison-Wesley, Longman Publishing Co., Inc., Reading, 2002.
[24] B. Boehm, A.W. Brown, R. Madachy, Y. Yang, A software product line cycle cost estimation model, in: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering, ISESE '04, IEEE Computer Society, 2004, pp. 156–164.
[25] W. Brace, K. Ekman, Coramod: a checklist-oriented model-based requirements analysis approach, Require. Eng. 19 (1) (2014) 1–26. http://dx.doi.org/10.1007/s00766-012-0154-3.
[26] M. Braz, S. Vergilio, Software effort estimation based on use cases, in: Proceedings of the 30th Annual International Computer Software and Applications Conference, COMPSAC '06, September 2006, pp. 221–228.
[27] L. Briand, Y. Labiche, A UML-based approach to system testing, Softw. Syst. Model. 1 (1) (2002) 10–42. http://dx.doi.org/10.1007/s10270-002-0004-8.
[28] W.J. Brown, R.C. Malveau, H.W. McCormick, III, T.J. Mowbray, Refactoring software, architectures, and projects in crisis, 1998.
[29] E.R. Carroll, Estimating software based on use case points, in: Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '05, ACM, NY, USA, 2005, pp. 257–265. http://doi.acm.org/10.1145/1094855.1094960.
[30] H. Christiansen, C.T. Have, K. Tveitane, From use cases to UML class diagrams using logic grammars and constraints, in: Proceedings of the International Conference on Recent Advances on Natural Language Processing, RANLP '07, pp. 128–132.
[31] A. Ciemniewska, J. Jurkiewicz, u. Olek, J. Nawrocki, Supporting use-case reviews, in: W. Abramowicz (Ed.), Business Information Systems, Lecture Notes in Computer Science, vol. 4439, Springer, Berlin Heidelberg, 2007, pp. 424–437. http://dx.doi.org/10.1007/978-3-540-72035-5_33.
[32] A. Cockburn, Writing Effective Use Cases, vol. 1, Addison-Wesley, Boston, 2001.
[33] K. Cox, A. Aurum, R. Jeffery, An experiment in inspecting the quality of use case descriptions, J. Res. Pract. Inf. Technol. 36 (4) (2004) 211–229.
[34] K. Cox, R. Jeffery, A. Aurum, A Use Case Description Inspection Experiment, University of New South Wales, School of Computer Science and Engineering, Technical report, 2004.
[35] K. Cox, K. Phalp, Replicating the crews use case authoring guidelines experiment, Emp. Softw. Eng. 5 (3) (2000) 245–267. http://dx.doi.org/10.1023/A%3A1026542700033.
[36] K. Cox, K. Phalp, Comparing use case writing guidelines, in: 7th International Workshop on Requirements Engineering: Foundation for Software Quality, 2001, pp. 101–112.
[37] K. Cox, K. Phalp, Exploiting use case descriptions for specification and design–an empirical study, in: Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering, Keele, UK, 2003, pp. 8–10.
[38] D.S. Cruzes, T. Dybå, Research synthesis in software engineering: a tertiary study, Inf. Softw. Technol. 53 (5) (2011) 440–455. special Section on Best Papers from {XP2010} <http://www.sciencedirect.com/science/article/pii/S095058491100005X>.
[39] J.C.S. Leite, G.D.S. Hadad, J.H. Doorn, G.N. Kaplan, A scenario construction process, Require. Eng. 5 (1) (2000) 38–61. http://dx.doi.org/10.1007/PL00010342.
[40] A.H. Dutoit, B. Paech, Rationale-based use case specification, Require. Eng. 7 (1) (2002) 3–19. http://dx.doi.org/10.1007/s007660200001.
[41] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (9) (2008) 833–859. <http://www.sciencedirect.com/science/article/pii/S0950584908000256>.
[42] T. Dybå, V.B. Kampenes, D.I. Sjøberg, A systematic review of statistical power in software engineering experiments, Inf. Softw. Technol. 48 (8) (2006) 745–755. <http://www.sciencedirect.com/science/article/pii/S0950584905001333>.
[43] T. Dybå, T. Dingsøyr, G.K. Hanssen, Applying systematic reviews to diverse study types: an experience report, in: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, ESEM '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 225–234. http://dx.doi.org/10.1109/ESEM.2007.21.
[44] M. El-Attar, Using SSUCD to develop consistent use case models: an industrial case study, in: Proceedings of the 7th International Conference on Software Engineering Advances, ICSEA 2012, 2012, pp. 172–178.
[45] M. El-Attar, J. Miller, AGADUC: towards a more precise presentation of functional requirement in use case models, in: Proceedings of the 4th International Conference on Software Engineering Research, Management and Applications, 2006, pp. 346–353.
[46] M. El-Attar, J. Miller, Producing robust use case diagrams via reverse engineering of use case descriptions, Softw. Syst. Model. 7 (1) (2008) 67–83. http://dx.doi.org/10.1007/s10270-006-0039-3.
[47] M. El-Attar, J. Miller, A subject-based empirical evaluation of SSUCDs performance in reducing inconsistencies in use case models, Emp. Softw. Eng. 14 (5) (2009) 477–512. http://dx.doi.org/10.1007/s10664-008-9101-9.
[48] M. El-Attar, J. Miller, Improving the quality of use case models using antipatterns, Softw. Syst. Model. 9 (2) (2010) 141–160. http://dx.doi.org/10.1007/s10270-009-0112-9.
[49] M. El-Attar, J. Miller, Constructing high quality use case models: a systematic review of current practices, Require. Eng. 17 (3) (2012) 187–201. http://dx.doi.org/10.1007/s00766-011-0135-y.
[50] S. Espana, N. Condori-Fernandez, A. Gonzalez, O. Pastor, Evaluating the completeness and granularity of functional requirements specifications: a controlled experiment, in: Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 161–170. http://dx.doi.org/10.1109/RE.2009.33.
[51] A. Fantechi, S. Gnesi, G. Lami, A. Maccari, Applications of linguistic techniques for use case analysis, Require. Eng. 8 (3) (2003) 161–170. http://dx.doi.org/10.1007/s00766-003-0174-0.
[52] W. Fleisch, Applying use cases for the requirements validation of component-based real-time software, in: Proceedings of the 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC '99, 1999, pp. 75–84.
[53] M. Fortuna, C. Werner, M. Borges, Info cases: integrating use cases and domain models, in: Proceedings of the 16th IEEE International Requirements Engineering, RE '08, 2008, pp. 81–84.
[54] P. Fröhlich, J. Link, Automated test case generation from dynamic models, in: Proceedings of the 14th European Conference on Object-Oriented Programming, ECOOP '00, Springer-Verlag, London, UK, UK, 2000, pp. 472–492. <http://dl.acm.org/citation.cfm?id=646157.679863>.
[55] P. Fusaro, F. Lanubile, G. Visaggio, A replicated experiment to assess requirements inspection techniques, Emp. Softw. Eng. 2 (1) (1997) 39–57. http://dx.doi.org/10.1023/A%3A1009742216007.
[56] J. Molina, M. Ortín, B. Moros, J. Nicolás, A. Toval, Towards use case and conceptual models through business modeling, in: A. Laender, S. Liddle, V. Storey (Eds.), Conceptual Modeling ER 2000, Lecture Notes in Computer Science, vol. 1920, Springer, Berlin Heidelberg, 2000, pp. 281–294. http://dx.doi.org/10.1007/3-540-45393-8_21.
[57] M.G. Georgiades, A.S. Andreou, Formalizing and automating use case model development, Open Softw. Eng. J. 6 (2012) 21–40.
[58] G. Génova, J. Fuentes, J. Llorens, O. Hurtado, V. Moreno, A framework to measure and improve the quality of textual requirements, Require. Eng. 18 (1) (2013) 25–41. http://dx.doi.org/10.1007/s00766-011-0134-z.
[59] G. Génova, J. Llorens, P. Metz, R. Prieto-Díaz, H. Astudillo, Open issues in industrial use case modeling, in: N. Jardim Nunes, B. Selic, A. Rodrigues da Silva, A. TovalAlvarez (Eds.), UML Modeling Languages and Applications, Lecture Notes in Computer Science, vol. 3297, Springer, Berlin Heidelberg, 2005, pp. 52–61. http://dx.doi.org/10.1007/978-3-540-31797-5_6.
[60] H. Gomaa, E.M. Olimpiew, The role of use cases in requirements and analysis modeling, in: Proceedings of the 2nd Interational Workshop on Use Case Modeling: Use Cases in Model-Driven Software Engineering, WUsCaM'05, 2005, pp. 2–7.
[61] I. Graham, Requirements Engineering and Rapid Development: An Object-Oriented Approach, ACM Press/Addison-Wesley Publishing Co, 1998.
[62] K.M. Hansen, A.P. Ravn, V. Stavridou, From safety analysis to software requirements, IEEE Trans. Softw. Eng. 24 (1998) 573–584.

[63] R.J. Harwood, Use case formats: requirements, analysis, and design, in: Journal of Object-Oriented Programming, 1997.
[64] P. Haumer, Use Case-Based Software Development, IBM Rational Rose, 2004, 1–27.
[65] J.H. Hausmann, R. Heckel, G. Taentzer, Detection of conflicting functional requirements in a use case-driven approach: A static analysis technique based on graph transformation, in: Proceedings of the 24th International Conference on Software Engineering, ICSE '02, ACM, New york, NY, USA, 2002, pp. 105–115. http://doi.acm.org/10.1145/581339.581355.
[66] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, R. Lutz, A software fault tree approach to requirements analysis of an intrusion detection system, Require. Eng. 7 (2002) 207–220, http://dx.doi.org/10.1007/s007660200016.
[67] J. Heumann <ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf> (accessed 02.05.14).
[68] R. Hurlbut, A Survey of Approaches for Describing and Formalizing Use Cases, Expertech, Ltd, 1997.
[69] R. Hurlbut, The Three Rs of Use Case Formalisms: Realization, Refinement, and Reification, 1997 <http://www.xltd.com/docs/xpt-tr-97-06.pdf>.
[70] M. Ilieva, O. Ormandjieva, Automatic transition of natural language software requirements specification into formal presentation, in: A. Montoyo, R. Munoz, E. Mtais (Eds.), Natural Language Processing and Information Systems, Lecture Notes in Computer Science, vol. 3513, Springer, Berlin Heidelberg, 2005, pp. 392–397. http://dx.doi.org/10.1007/11428817_45.
[71] E. Insfŕan, O. Pastor, R. Wieringa, Requirements engineering-based conceptual modelling, Require. Eng. 7 (2) (2002) 61–72. http://dx.doi.org/10.1007/s007660200005.
[72] ISO9126, Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use, 1992.
[73] A. Issa, M. Odeh, D. Coward, Using use case patterns to estimate reusability in software systems, Inf. Softw. Technol. 48 (9) (2006) 836–845. special Issue Section: Distributed Software Development <http://www.sciencedirect.com/science/article/pii/S0950584905001588>.
[74] I. Jacobson, M. Christerson, P. Jonsson, G. Overgaard, Object-Oriented Software Engineering: A Use-Case Driven Approach, Addison-Wesley, Reading, MA, 1992. 1992 Edition.
[75] M. Ivarsson, T. Gorschek, Technology transfer decision support in requirements engineering research: a systematic review of REJ, Require. Eng. 14 (3) (2009) 155–175.
[76] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Emp. Softw. Eng. 16 (3) (2011) 365–395. http://dx.doi.org/10.1007/s10664-010-9146-4.
[77] A. Jaaksi, Our cases with use cases, in: Journal of Object-Oriented Programming, 1998.
[78] I. Jacobson, Object-oriented development in an industrial environment, in: Proceedings of the Conference on Object-Oriented Programming, Systems, Languages & Applications, ACM, 1987, pp. 183–191.
[79] I. Jacobson, Use cases – yesterday, today, and tomorrow, Softw. Syst. Model. 3 (3) (2004) 210–220. http://dx.doi.org/10.1007/s10270-004-0060-3.
[80] M. Kamal, M. Ahmed, M. El-Attar, Use case-based effort estimation approaches: a comparison criteria, in: J. Zain, W. Wan Mohd, E. El-Qawasmeh (Eds.), Software Engineering and Computer Systems, Communications in Computer and Information Science, vol. 181, Springer, Berlin Heidelberg, 2011, pp. 735–754. http://dx.doi.org/10.1007/978-3-642-22203-0_62.
[81] E. Kamsties, A. von Knethen, R. Reussner, A controlled experiment to evaluate how styles affect the understandability of requirements specifications, Inf. Softw. Technol. 45 (14) (2003) 955–965. eighth International Workshop on Requirements Engineering: Foundation for Software Quality <http://www.sciencedirect.com/science/article/pii/S0950584903000983>.
[82] J. Kanyaru, K. Phalp, Validating software requirements with enactable use case descriptions, Require. Eng. 14 (1) (2009) 1–14. http://dx.doi.org/10.1007/s00766-008-0070-8.
[83] J. Kanyaru, K. Phalp, K. Cox, A. Jha, Supporting the consideration of dependencies in use case specifications, in: Proceedings of the 11th International Workshop on Requirements Engineering: Foundation For Software Quality, REFSQ05 2005, pp. 13–14.
[84] G. Karner, Resource estimation for objectory projects, in: Objective Systems SF AB (copyright owned by Rational Software), 1993.
[85] S. Keele, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Tech. rep., Technical report, EBSE Technical Report EBSE-2007-01, 2007.
[86] J. Kettenis, Getting Started With Use Case Modeling: White Paper, Oracle Corporation, USA, 2007.
[87] M.T. Kimour, D. Meslati, Deriving objects from use cases in real-time embedded systems, Inf. Softw. Technol. 47 (8) (2005) 533–541. http://dx.doi.org/10.1016/j.infsof.2004.10.003.
[88] B. Kitchenham, Procedures for Performing Systematic Reviews, Keele, UK, Keele University 33, 2004.
[89] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering a systematic literature review, Inf. Softw. Technol. 51 (1) (2009) 7–15. special Section - Most Cited Articles in 2002 and Regular Research Papers <http://www.sciencedirect.com/science/article/pii/S0950584908001390>.
[90] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Inf. Softw. Technol. 55 (12) (2013) 2049–2075. <http://www.sciencedirect.com/science/article/pii/S0950584913001560>.
[91] B. Kitchenham, S.L. Pfleeger, Software quality: the elusive target, IEEE Softw. 13 (1) (1996) 12–21.
[92] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley, Boston, 2003.
[93] D. Kulak, E. Guiney, Use Cases: Requirements in Context, Addison-Wesley, 2012.
[94] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3/e, Pearson Education India, 2012.
[95] S. Lauesen, Software Requirements: Styles and Techniques, Pearson Education, 2002.
[96] J. Lee, N.L. Xue, Analyzing user requirements by use cases: a goal-driven approach, IEEE Softw. 16 (4) (1999) 92–101.
[97] S. Lilly, Use case pitfalls: top 10 problems from real projects using use cases, in: Proceedings of the Technology of Object-Oriented Languages and Systems, TOOLS '99, 1999, pp. 174–183.
[98] D. Liu, K. Subramaniam, B. Far, A. Eberlein, Automating transition from use-cases to class model, in: Proceedings of the Canadian Conference on Electrical and Computer Engineering, IEEE CCECE 2003, vol. 2, 2003, pp. 831–834.
[99] L. Mattingly, H. Rao, Writing effective use cases and introducing collaboration cases, in: Journal of Object-Oriented Programming, 1998.
[100] J.R. McCoy, Requirements use case tool (RUT), in: Companion Proceedings of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '03, ACM, NY, USA, 2003, pp. 104–105. http://doi.acm.org/10.1145/949344.949371.
[101] J. Miller, M. Wood, M. Roper, Further experiences with scenarios and checklists, Emp. Softw. Eng. 3 (1) (1998) 37–64. http://dx.doi.org/10.1023/A%3A1009735805377.
[102] M. Misbhauddin, M. Alshayeb, Extending the uml use case metamodel with behavioral information to facilitate model analysis and interchange, Softw. Syst. Model, 2013, 1–26, http://dx.doi.org/10.1007/s10270-013-0333-9.
[103] P. Mohagheghi, B. Anda, R. Conradi, Effort estimation of use cases for incremental large-scale software development, in: Proceedings of the 27th International Conference on Software Engineering, ICSE '05, ACM, NY, USA, 2005, pp. 303–311. http://doi.acm.org/10.1145/1062455.1062516.
[104] C. Nebut, F. Fleurey, y. Le Traon, J.-M. Jezequel, Automatic test generation: a use case driven approach, IEEE Trans. Softw. Eng. 32 (3) (2006) 140–155.
[105] M. Ochodek, J. Nawrocki, K. Kwarciak, Simplifying effort estimation based on use case points, Inf. Softw. Technol. 53 (3) (2011) 200–213. http://dx.doi.org/10.1016/j.infsof.2010.10.005.
[106] J. Offutt, A. Abdurazik, Generating tests from uml specifications, in: Proceedings of the 2nd International Conference On The Unified Modeling Language: Beyond the Standard, UML'99, Springer-Verlag, Berlin, Heidelberg, 1999, pp. 416–429.
[107] G. Övergaard, K. Palmkvist, A formal approach to use cases and their relationships, in: The Unified Modeling Language, UML'98: Beyond the Notation, Springer, 1999, pp. 406–418.
[108] G. Övergraad, K. Palmkvist, Use Cases Patterns and Blueprints, Addison-Wesley, 2005.
[109] D. Parachuri, A.S.M. Sajeev, R. Shukla, An empirical study of structural defects in industrial use-cases, in: Companion Proceedings of the 36th International Conference on Software Engineering, ICSE Companion 2014, ACM, New York, NY, USA, 2014, pp. 14–23. http://doi.acm.org/10.1145/2591062.2591174.
[110] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, British Computer Society, Swinton, UK, 2008, pp. 68–77 <http://dl.acm.org/citation.cfm?id=2227115.2227123>.
[111] K. Phalp, K. Cox, Guiding use case driven requirements elicitation and analysis, in: X. Wang, R. Johnston, S. Patel (Eds.), OOIS 2001, Springer, London, 2001, pp. 329–332. http://dx.doi.org/10.1007/978-1-4471-0719-4_34.
[112] K. Phalp, K. Cox, Supporting communicability with use case guidelines: an empirical study, in: Proceedings of 6th International Conference on Empirical Assessment in Software Engineering, April 2002, pp. 8–10.
[113] K. Phalp, K. Cox, Using enactable models to enhance use case descriptions, in: Proceedings of the ProSim 03, International Workshop on Software Process Simulation Modelling (in Conjunction with ICSE), 2003, Portland, USA.
[114] K.T. Phalp, J. Vincent, K. Cox, Assessing the quality of use case descriptions, Softw. Qual. J. 15 (1) (2007) 69–97. http://dx.doi.org/10.1007/s11219-006-9006-z.
[115] K.T. Phalp, J. Vincent, K. Cox, Improving the quality of use case descriptions: empirical assessment of writing guidelines, Softw. Qual. Control 15 (4) (2007) 383–399. http://dx.doi.org/10.1007/s11219-007-9023-6.
[116] A. Porter, L.G. Votta, V. Basili, Comparing detection methods for software requirements inspections: a replicated experiment, IEEE Trans. Softw. Eng. 21 (6) (1995) 563–575.
[117] R.S. Pressman, in: Software Engineering: A Pratitioner's Approach, seventh ed., McGraw-Hill Higher Education, 2009.
[118] A. Rago, C. Marcos, J. Diaz-Pace, Uncovering quality-attribute concerns in use case specifications via early aspect mining, Require. Eng. 18 (1) (2013) 67–84. http://dx.doi.org/10.1007/s00766-011-0142-z.

[119] M. Ratcliffe, D. Budgen, The application of use cases in systems analysis and design specification, Inf. Softw. Technol. 47 (9) (2005) 623–641. <http://www.sciencedirect.com/science/article/pii/S0950584904001727>.

[120] B. Regnell, K. Kimbler, A. Wesslen, Improving the use case driven approach to requirements engineering, in: Proceedings of the 2nd IEEE International Symposium on Requirements Engineering, 1995, March 1995, pp. 40–47.

[121] A. Rodríguez, I.G.-R. d. Guzmán, E. Fernández-Medina, M. Piattini, Semi-formal transformation of secure business processes into analysis class and use case models: an MDA approach, Inf. Softw. Technol. 52 (9) (2010) 945–971. http://dx.doi.org/10.1016/j.infsof.2010.03.015.

[122] C. Rolland, C.B. Achour, Guiding the construction of textual use case specifications, Data Knowl. Eng. 25 (12) (1998) 125–160. <http://www.sciencedirect.com/science/article/pii/S0169023X97862234>.

[123] K. Rui, G. Butler, Refactoring use case models: the metamodel, in: Proceedings of the 26th Australasian Computer Science Conference – vol. 16, ACSC '03, Australian Computer Society, 2003, pp. 301–308. <http://dl.acm.org/citation.cfm?id=783106.783140>.

[124] M. Saeki, Reusing use case descriptions for requirements specification: towards use case patterns, in: Proceedings of the 6th Asia Pacific Software Engineering Conference APSEC '99, 1999, pp. 309–316.

[125] V.F.A. Santander, J. Castro, Deriving use cases from organizational modeling, in: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering, RE '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 32–42. <http://dl.acm.org/citation.cfm?id=647648.731618>.

[126] D. Savic, I. Antovic, S. Vlajic, V. Stanojevic, M. Milic, Language for use case specification, in: Proceedings of the 34th IEEE Software Engineering Workshop, SEW, June 2011, pp. 19–26.

[127] G. Schneider, J. Winters, Applying Use Cases A Practical Guide, Addison-Wesley, 1998.

[128] K. Siau, L. Lee, Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in uml, Require. Eng. 9 (4) (2004) 229–237. http://dx.doi.org/10.1007/s00766-004-0203-7.

[129] A. Sinha, Jr., S.M. S, A. Paradkar, Text2test: automated inspection of natural language use cases, in: Proceedings of the 2013 IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2010, 2010, 155–164.

[130] A. Sinha, A. Paradkar, P. Kumanan, B. Boguraev, A linguistic analysis engine for natural language use case description and its application to dependability analysis in industrial use cases, in: Proceedings of the IEEE/IFIP International Conference on Dependable Systems Networks, DSN '09, June 2009, pp. 327–336.

[131] S.S. Somé, Supporting use case based requirements engineering, Inf. Softw. Technol. 48 (1) (2006) 43–58. <http://www.sciencedirect.com/science/article/pii/S0950584905000285>.

[132] S.S. Somé, Formalization of textual use cases based on petri nets, Int. J. Softw. Eng. Knowl. Eng. 20 (05) (2010) 695–737.

[133] I. Sommerville, P. Sawyer, Requirements Engineering: A Good Practice Guide, John Wiley & Sons, Inc., 1997.

[134] OMG Specifications, Big Picture <http://www.omg.org/> (accessed 02.07.12).

[135] K. Subramaniam, D. Liu, B.H. Far, A. Eberlein, UCDA: use case driven development assistant, 2004.

[136] E. Syversen, B. Anda, D. Sjøberg, An evaluation of applying use cases to construct design versus validate design, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, January 2003, pp. 10–10.

[137] S. Tena, D. Dez, P. Daz, I. Aedo, Standardizing the narrative of use cases: a controlled vocabulary of web user tasks, Inf. Softw. Technol. 55 (9) (2013) 1580–1589. <http://www.sciencedirect.com/science/article/pii/S0950584913000451>.

[138] T. Thelin, P. Runeson, C. Wohlin, An experimental comparison of usage-based and checklist-based reading, IEEE Trans. Softw. Eng. 29 (8) (2003) 687–704.

[139] S. Tiwari, A. Gupta, A controlled experiment to assess the effectiveness of eight use case templates, in: Proceedings of the 20th Asia–Pacific Software Engineering Conference, APSEC 2013, December 2013, pp. 207–214.

[140] S. Tiwari, A. Gupta, Does increasing formalism in the use case template help?, in: Proceedings of the 7th India Software Engineering Conference, ISEC 2014, ACM, NY, USA, 2014, pp 6:1–6:10. http://doi.acm.org/10.1145/2590748.2590754.

[141] S. Tiwari, S. Rathore, S. Gupta, V. Gogate, A. Gupta, Analysis of use case requirements using SFTA and SFMEA techniques, in: Proceedings of the 17th International Conference on Engineering of Complex Computer Systems, ICECCS 2012, July 2012, pp. 29–38.

[142] F. Tormer, M. Ivarsson, F. Pattersson, An empirical quality assessment of automotive use cases, in: Proceedings of the 21st IEEE International Requirements Engineering Conference, RE 2013, 2013, pp. 89–98.

[143] A.D. Toro, B.B. Jiménez, A.R. Cortés, M.T. Bonilla, A requirements elicitation approach based in templates and patterns, in: WER, 1999, pp. 17–29.

[144] K. Van den Berg, A. Simons, Control-flow semantics of use cases in UML, Inf. Softw. Technol. 41 (10) (1999) 651–659. <http://www.sciencedirect.com/science/article/pii/S0950584999000270>.

[145] J. Whittle, Precise specification of use case scenarios, in: M. Dwyer, A. Lopes (Eds.), Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science, vol. 4422, Springer, Berlin Heidelberg, 2007, pp. 170–184. http://dx.doi.org/10.1007/978-3-540-71289-3_15.

[146] K.E. Wiegers, Writing quality requirements, Softw. Develop. 7 (5) (1999) 44–48.

[147] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: A proposal and a discussion, Require. Eng. 11 (1) (2005) 102–107. http://dx.doi.org/10.1007/s00766-005-0021-6.

[148] Y. Liang, From use cases to classes: a way of building object model with UML, Inf. Softw. Technol. 45 (2) (2003) 83–93. <http://www.sciencedirect.com/science/article/pii/S0950584902001647>.

[149] T. Yue, L. Briand, y. Labiche, An automated approach to transform use cases into activity diagrams, in: T. Khne, B. Selic, M.-P. Gervais, F. Terrier (Eds.), Modelling Foundations and Applications, Lecture Notes in Computer Science, vol. 6138, Springer, Berlin Heidelberg, 2010, pp. 337–353. http://dx.doi.org/10.1007/978-3-642-13595-8_26.

[150] T. Yue, L. Briand, y. Labiche, A systematic review of transformation approaches between user requirements and analysis models, Require. Eng. 16 (2) (2011) 75–99. http://dx.doi.org/10.1007/s00766-010-0111-y.

[151] T. Yue, L.C. Briand, y. Labiche, Facilitating the transition from use case models to analysis models: Approach and experiments, ACM Trans. Softw. Eng. Methodol. 22 (1) (2013) 5:1–5:38. http://doi.acm.org/10.1145/2430536.2430539.