The Consolidation of Game Software Engineering: A Systematic Literature Review of Software Engineering for Industry-scale Computer Games

Jorge Chueca[a], Javier Verón[a], Jaime Font[a], Francisca Pérez[a], Carlos Cetina[a,b]

[a] *SVIT Research Group, Universidad San Jorge, Zaragoza, Spain*
[b] *University College London, London, United Kingdom*

Abstract

Context: Game Software Engineering (GSE) is a branch of Software Engineering (SE) that focuses on the development of video game applications. In past years, GSE has achieved enough volume, differences from traditional software engineering, and interest by the community to be considered an independent scientific domain, veering out from traditional SE.

Objective: This study evaluates the current state of the art in software engineering for industry-scale computer games identifying gaps and consolidating the magnitude and growth of this field.

Method: A Systematic Literature Review is performed following best practices to ensure the relevance of the studies included in the review. We analyzed 98 GSE studies to extract the current intensity, topics, methods, and quality of GSE.

Results: The GSE research community has been growing over the years, producing over four times more research than before the previous GSE survey. However, this community is still very dispersed, with no main venues holding most of the GSE scientific studies. A broader range of topics is covered in this area, evolving towards those of a mature field such as architecture and design. Also, the reviewed studies employ more elaborated empirical research methods, even though the study reports need to be more rigorous in sections related to the critical examination of the work.

Conclusion: The results of the SLR lead to the identification of 13 potential future research directions for this domain. GSE is an independent, mature, and growing field that presents new ways of software creation where the gap between industry and academia is narrowing. Video games present themselves as powerful tools to push the boundaries of software knowledge.

Keywords: Computer Games, Video Games, Game Software Engineering, Systematic Literature Review, SLR, industry-scale.

1. Introduction

In 2021, the video game industry generated total revenues of $180.3 billion (up 1.4% over 2020 [1]), and it keeps growing after having already surpassed the revenues of the movie and music industries combined in 2019 [2], this makes it the largest entertainment industry in terms of revenue.

Video games are advanced software products and, at the same time, they are complex works of creativity and art [3]. This merge of disciplines, the need for the product to be fun, and the unrelenting growth of the video game industry have created significant differences between traditional Software Engineering (SE) and Game Software Engineering (GSE) [4]. These conditions require constant and fast communication among developers, flexible design, and maintainable implementation. Moreover, video games tend to be products with a much more limited life cycle than other software products [5]. In Section 2, we present more differences and similarities between GSE and traditional SE, showing that the differences outnumber the similarities. For these reasons, we propose that GSE be considered an independent domain that presents divergence from traditional software engineering.

Email addresses: jchueca@usj.es (Jorge Chueca), jveron@usj.es (Javier Verón), jfont@usj.es (Jaime Font), mfperez@usj.es (Francisca Pérez), ccetina@usj.es (Carlos Cetina)

1.1. Goals and Contributions

Many authors [6, 7, 4, 5] argue that GSE and traditional SE differ more than they might be currently conceived. We aim to show whether or not GSE has enough volume and interest by the community to be considered a whole domain instead of a small software engineering subtopic.

In addition, we also want to shed light on the current state of research on software engineering techniques, approaches, and methods being applied to computer games and to compare their evolution before and after 2009 (last year covered by the previous GSE survey [5]), in order to provide background information for any relevant GSE work in the future. This background information will also benefit both experts and newcomers to this field by showing the tendencies and gaps in current research. Furthermore, the results of the survey lead to the identification of several potential future research directions for GSE.

Finally, this study tightens the gap between industry and academia, showing both of them how to nourish each other. We present a large set of industry-scale research that invites practitioners to publish their expertise in research venues, and we show SE researchers that GSE is a fertile domain with plenty of case studies to build up the knowledge of software.

1.2. Lack of Existing SLRs

Some authors have reviewed different topics of video games as a software domain. However, there is a dearth of systematic reviews on more general topics such as Software Engineering. The three main approaches to reviewing the topic of video games as software have been **surveys and interviews** (six studies), **grey literature** (seven studies), and **academic literature** (nine studies). Table 1 shows the studies that we found about GSE, followed by a brief commentary of each study.

Table 1: Scientific reports on GSE.

Approach	Studies	Total
Surveys and Interviews	Murphy et al. [8] Kasurinen et al. [9] Nandhakumar et al. [10] Kasurinen et al. [6] McKenzie et al. [11] Musil et al. [12]	6
Grey Literature Reviews	Politowski et al. [13] Politowski et al. [14] Politowski et al. [15] Politowski et al. [16] Ullmann et al. [17] Petrillo et al. [18] Washburn et al. [19]	7
Academic Literature Reviews	Engström et al. [3] Mizutani et al. [20] Viana et al. [21] Zhu et al. [22] Politowski et al. [23] Osborne et al. [24] Vargas et al. [25] Aleem et al. [26] **Ampatzoglou et al. [5]**	9

The **surveys and interviews** that deal with the domain of games as software are the following: Murphy et al. [8] surveyed 14 game and non-game developers, finding different practices among these domains. Kasurinen et al. [9] conducted 27 interviews with professional game developers to understand how they test their products. Nandhakumar et al. [10] surveyed and observed three different development companies to explore the challenges of conceptualizing and realizing the desired user experience. Kasurinen et al. [6] interviewed 33 professionals from different game development companies, focusing on the particular characteristics of video game business topics. McKenzie et al. [11] interviewed 12 participants from New Zealand's game industry, revealing that custom agile methodologies are the leading practice for developers. Musil et al. [12] created a questionnaire for Austrian game studios relating scope and industry-specific problems.

Some contributions are only reported in the **grey literature** and at professional venues such as the Game Developers Conference. Politowski et al. have surveyed postmortem project analyzes from the grey literature to find what software engineering processes are being used in the video game industry [13] and the main development problems in the current industry [14]. They have also surveyed grey literature on existing testing processes [15]. Politowsky et al. created a dataset of more than 200 postmortems gathering 1,035 problems related to software engineering in video games [16]. Ullmann et al. read 440 postmortem problems to identify anti-patterns in game development [17]. Petrillo et al. [18] examined 20 postmortems to find how agile methodologies are applied in video game development and how the lack of a rigorous methodology is related to an increase in problems found in development. Washburn et al. [19] analyzed 155 postmortems, finding best practices and pitfalls in game development.

Other researchers have sought answers by reviewing **academic literature**. Engström et al. [3] conducted a SLR, bridging the gap between the creative and technical nature of video games. Mizutani et al. [20] performed a SLR on the usage of software architectures in game mechanics. Viana et al. [21] carried out a SLR on Software Engineering for Pervasive Games. Zhu and Wang [22] analyzed the state of the art of Model-Driven Game Development (MDGD) research with a literature review. Politowski et al. [23] performed a literature review and a survey of developers about the topic of automated video game testing. Osborne et al. [24] conducted a SLR about processes and management in GSE research. Vargas et al. [25] performed a mapping study to reveal the growth in the quality of serious games. Aleem et al. [26] carried out a SLR on the production life cycle of video games.

All of the previous academic literature reviews focus on specific topics within GSE. To the best of our knowledge, the only study that addresses software and its engineering in video games is the SLR published by Ampatzoglou and Stamelos in 2010 [5], which is highlighted in bold in Table 1. They confirmed that GSE is *"a fertile domain"* with high activity, intensity growth, and open avenues to continue and validate their study with future research.

This study consolidates the hypothesis given by Ampatzoglou and Stamelos [5], providing an updated perspective on the state and evolution of GSE. The results of their study and ours are compared, rendering the state of GSE from its beginning to the present. The difference between our study and previous research is that our study explores the last decade of GSE, which has never been researched before, and it also focuses on industry-scale evidence.

1.3. Structure of this work

The remainder of this study report is structured as follows: Section 2 displays some background on video games as software and presents the previous related work. Section 3 addresses the research questions and the study design, explaining the review method step by step as we performed it. Section 4 presents the results we obtained as the data extracted from the reviewed studies answering the research questions. Section 5 presents a discussion of those results. Section 6 looks forward into the future of GSE. Section 7 presents possible threats to validity, limitations, and a quality assessment of this review.

Finally, Section 8 concludes the study report and identifies the next steps for future work.

2. Background

Video games are software and, thus, present similarities with methods and techniques that are present in Software Engineering. Nevertheless, video games are a unique piece of software in terms of size, complexity, and creativity, constituting plenty of differences between GSE and traditional SE. As we performed the literature review, we discovered how the research community earmarked efforts to determine the nature of video games as software, finding their similarities and differences with traditional software.

We performed a manual search of the research in order to find the similarities and differences between these two fields. We found that there are four main similarities shared between GSE and traditional SE:

- The design decisions are based on a commercial perspective, with priority on the business side [7, 26, 27, 28, 29].
- Technical work is similar between the fields. Even if the requirements are specific for each domain, the programming work must adhere to those requirements [2, 7, 27].
- Both face similar problems in project scope, dropping features, and design. [7, 10, 27, 28, 30, 31].
- Despite potential disparities due to team size and role heterogeneity, project management exhibits similarity between the two fields, as game development teams have adopted agile methodologies from traditional software engineering [3, 4, 7, 31, 32].

On the other hand, the differences we found have a critical impact on the approach of video games towards software. The studies concerning GSE that tackle the differences have ten points in common:

- The teams developing a video game are multidisciplinary, with non-engineer heterogeneous roles (e.g., art, music, animation, or marketing) that must be integrated into the work pipeline [31, 6, 8, 3, 12, 33, 18, 26, 10].
- The process of game development is highly iterative, and the design is made while implementing the software itself [31, 7, 4, 3, 12]. This leads to a dearth of non-agile methodologies for video game development [6, 8, 11, 18].
- Video game developers usually have fixed deadlines in a market where delaying a release can impact the business considerably [5, 12].
- Video game developers must manage a large number of assets since customers expect large amounts of content, mirroring the industry's progression from smaller, technologically limited games to the current trend of expansive, content-rich gaming experiences [31, 6, 7].
- Video games also place emphasis on player experience testing and game-specific testing. This emphasis on player experience takes into account the interconnection of aesthetics, interactive mechanics, and storytelling. Game-specific testing encompasses comprehensive gameplay evaluation, including controls, mechanics, and level design assessments, such as balancing. Not only must game developers perform technical testing, but they also need to test the artistic nature of their products. [6, 7, 33, 9].
- Post-release features and content addition are standard practices in the video game industry. The product's life cycle can be extended years after launch [7, 33].
- Video games have non-functional endemic requirements that are highly subjective and more focused on player experience [33, 7, 32, 26, 10]. These user-driven requirements cannot be described technically since they are based on the purely subjective opinion of the users and focus on abstract concepts such as "*fun*" [4], "*entertainment*", "*aesthetics*", "*flow*" [32], or "*creativity*" [33] instead of being based on real-world and measurable concepts as in traditional software [7].
- Video games require the management of complex software systems that must communicate and synchronize with each other for an understandable and enjoyable experience. These systems include 2D & 3D rendering, physics, sound, and artificial intelligence [7, 12, 26].
- The complexity of video games usually forces game developers to use sophisticated tools called "game engines" that require specific expertise. These engines require skills other than software frameworks, such as art, design, or musical skills. These engines are also used differently than frameworks. To ease the design and implementation process, the game developer often uses scripting languages or domain-specific modeling languages that are different from the language the engine is coded [7, 2, 8, 32].

While GSE shares some traditional practices with SE, many other practices differ, and there is a need for specificity while dealing with video game software. These facts bring us to the same conclusion that many other researchers have come to previously [6, 7, 4]: *Game Software Engineering is a scientific domain rather than a soft skill topic with only a peripheral research activity* [5].

3. Review Method

The systematic review process details how to analyze and synthesize the evidence addressed by other scientific studies related to the subject in an unbiased and repeatable way. We followed the guidelines given by Kitchenham and Charters [34]. The stages of a systematic review are divided into three main phases: Planning the review, Conducting the review, and Reporting the review [34, 35]. Every phase is a combination of other more straightforward procedures: The Planning phase

combines the specification of research questions and the development and validation of the review protocol. The Conducting phase requires the identification of relevant research, the selection of the primary studies, the assessment of the studies' quality, the extraction of the required data, and the synthesis of the data. Finally, the Reporting phase is performed by writing the review report and validating it. Following the best practices [36, 5] to plan our review, we define the following six elements: research questions, search process, inclusion and exclusion criteria, quality assessment, data collection, and data analysis.

The first and second authors participated in every step of the review process, double-checking every step to ensure the accuracy of the inclusion and exclusion processes and the quality assessment. When disagreements occurred, the two reviewers discussed the possibilities. If both reviewers did not reach a consensus, the other authors resolved the dichotomy.

3.1. Research Questions

Alves et al. [37] use a classification approach for the maturity of a particular method or tool. This is a revision of Kitchenham's levels of study design in software engineering based on the evidence hierarchy [38]. This classification's strongest ranks are *evidence obtained from industrial studies* and *evidence obtained from industrial practice*. The rating *"industrial practice"* indicates that the method has already been approved and adopted by some organizations. Such daily engineering practice provides convincing proof of the study's validity [37]. By selecting only studies with industrial relations, we ensure that the quality of our provided results matches the high standards of the industry.

This research field is growing actively year by year, and thus in quality and maturity [5]. Because of that, we have the opportunity to specify the systematic review to focus on software engineering for **industry-scale** computer games. The term "industry-scale computer game" refers to games and tools developed in industrial practice or for commercial purposes, thus, we do not consider academic prototypes as industry-scale computer games. This study addresses similar research questions (RQ1, RQ2, RQ3, and RQ4) from Ampatzoglou and Stamelos' [5] study with adaptations to our more restrictive research domain to allow for a more straightforward comparison.

RQ1: Has the intensity of the research activity on software engineering for industry-scale computer game development changed over the years?

RQ2: How have software engineering research topics in the domain of industry-scale computer games changed over the years?

RQ3: How have research approaches in the domain of industry-scale computer games changed over the years?

RQ4: How have empirical research methods in the domain of industry-scale computer games changed over the years?

RQ5: What is the quality of industry-scale computer game research currently?

RQ1 shows the existence of an increase or a decrease in research activity. We compared the variation for research in software engineering in computer games since 2009 to the corresponding variation of the years prior to 2009, as provided by [5].

To address RQ2, RQ3, and RQ4, we used the data gathered in the review and also compared it with the current research study on the topic by Ampatzoglou and Stamelos [5] prior to 2009. We associated each primary study with a research topic, a specific research approach, and a specific research empirical methodology, similar to what Ampatzoglou and Stamelos did. For each RQ, we add a discussion drawing conclusions from the additional evidence obtained, shedding some light on the current state of GSE and identifying future trends.

With regard to RQ2, the classification system of the primary studies that [5] employed is the ACM Computing Classification System, which was widely adopted in computing-related engineering research. The ACM CCS was updated in 2012 [39], replacing the previous 1998 version. We classified the primary studies with this new version, but we also classified them with the 1998 version in order to make the comparison. These classifications are further explained in Section 3.4.1.

With regard to RQ3, scientific research studies can be classified with respect to their approach [40]. According to the classification scheme proposed by R.L. Glass et al. [40] and later on used by Ampatzoglou and Stamelos [5], scientific studies can be categorized by the overall approach undertaken in performing the research. The main scientific approach categories are descriptive, exploratory, and empirical approaches [5]. Descriptive studies describe a system, tool, or method. Moreover, SLRs are considered to be descriptive. Exploratory studies determine the best research design, data collection method, and selection of subjects. The findings of empirical studies are based on direct or indirect observation of real subjects.

With regard to RQ4, empirical research studies can be classified with respect to the method of empirical investigation used to evaluate new techniques, methods, and tools [41]. These method categories are surveys, case studies, and experiments. Experiments have a set of subjects performing a task in a highly controlled environment. The results are obtained by observing the subjects during the experiment, inspecting the task outcome, or questioning the subjects after performing the task. Surveys are usually used if the method under study has been in use for a while. In surveys, a set of subjects is asked to fill in questionnaires. The results are obtained with valid answers to these questionnaires. Finally, case studies study a methodology by monitoring a project, activity, or assignment with respect to it. The results are obtained with project measurements [42].

The same classification of approaches and empirical methods was employed in [5], so we can compare the results in that study with our own and see an evolution of the results before and after 2009.

RQ5 allows us to identify the common problems in current research that are tampering with the quality of the studies. This will help researchers to avoid those issues in future work in the field of GSE.

3.2. Literature Search Strategy

We performed a database search to gather all of the studies that were relevant to the research questions. To do this, we performed the following steps: search term extraction, database selection, and selection process.

3.2.1. Search terms

We used PICO (Population, Intervention, Comparison, Outcome) as suggested by Kitchenham and Charters [34] to find the search terms for the query.

Population. In software engineering, population refers to the application area in software engineering [34]. For our study, this is computer games.

Intervention. In software engineering, intervention refers to the methodologies, tools, technologies, or procedures that are used to create the software [34]. In our study, this corresponds to 'engineering', 'development', 'requirements', 'design', 'coding', 'testing', 'verification', 'evolution', and 'maintenance'. We have extracted these terms to describe the different software engineering activities, comparable to how Mao et al. [43] extracted the terms for a similar survey reviewing another broad SE field.

Comparison. In software engineering, comparison refers to the methodologies, tools, technologies, or procedures being compared with the intervention [34]. In our study, comparison is not applicable.

Outcome. In software engineering, outcome refers to factors that should be important to practitioners [34]. In our study, the outcome includes approaches that focus on industry-scale video games and Software Engineering. We decided to perform this step manually as part of the inclusion and exclusion criteria to avoid losing valuable work if it is filtered automatically. There are many studies performed in industry-grade video games where the word "industrial" does not appear in the whole study report.

After recognizing the search terms from the research questions, these terms need modifications and additions to be effective in a search string.

The term *computer game* refers to electronic games that are developed to be executed on a computer. These games can be nominated as *video games*, an alternative synonym that refers to games that are developed to be executed on any device that supports them, including computers. Even though it is a more generic term, we included it to extract studies deemed significant. Since we only intended to review computer games, we make up for this addition with the selection criteria, which is detailed in Section 3.2.4. With this criteria, we exclude any study about non-computer games. *Video games* can also be spelled as *videogames*. Additionally, some databases need the distinction between plural and singular words, so we added these variations to the search string.

Our Intervention terms can also refer to ideas that are too broad and can depict concepts outside of the SE field, so we added the word *software* before each term. For example, the term *testing* can be used in the medical field, and, thus, it is changed to *software testing* to ensure its relevance to SE.

For the search string, we used the Boolean operator OR to combine the terms inside each criterion and the Boolean operator AND between Population and Intervention. The terms of Population must appear in the title, abstract, or keywords of the searched studies to ensure the focus on computer game development. The terms of Intervention must appear anywhere in the studies found. The resulting search string is as follows:

TITLE-ABS-KEY ("computer game" OR "computer games" OR "video game" OR "video games" OR "videogame" OR "videogames") AND ALL ("software engineering" OR "software development" OR "software requirements" OR "software design" OR "software coding" OR "software testing" OR "software verification" OR "software evolution" OR "software maintenance")

3.2.2. Databases

We have considered the Scopus, Web of Science, IEEE, and ACM databases for this survey. We chose these based on the prior experience of Dyba et al. [44] and Kitchenham and Brereton [45]. These studies stated that using IEEE and ACM as well as two indexing databases is appropriate [46]. During the search, we considered all of the full years from 2009 to 2021 (both included). Fig. 1 shows the number of search results per database.

3.2.3. Search process

Both reviewers followed the review and selection process depicted in Fig. 1 using the 3,491 total results from the search in the four databases. To achieve the selection of studies for data extraction, the process performed had the following steps:

1. The reviewers searched four databases using the search string to collect the primary studies. We show the search results from each database in Fig. 1.

2. The total number of 3,491 studies was filtered to exclude 517 duplicate records, decreasing it to 2,974 studies.

3. The non-duplicated studies were filtered, and we excluded all non-relevant studies for this study. For the fulfilment of this step, we applied the exclusion criteria considering each study's title and keywords. The selection decreased to 614 studies, which we selected for the following step.

4. We filtered the 614 studies. This time, we filtered the studies by abstract with the same exclusion criteria, and 362 were eliminated. The selection decreased to 252 studies not meeting any exclusion criterion.

5. We also filtered this selection of 252 studies by their relevance in answering the research questions, applying the inclusion criteria to the full text of each selected study. The selection again decreased to a final selection of 90 studies.

6. We selected all of the references from the selection of 90 studies for the application of backward snowballing [47]. We performed snowballing to search for possible missing studies. The initial list of snowballing studies counted was 591 studies.

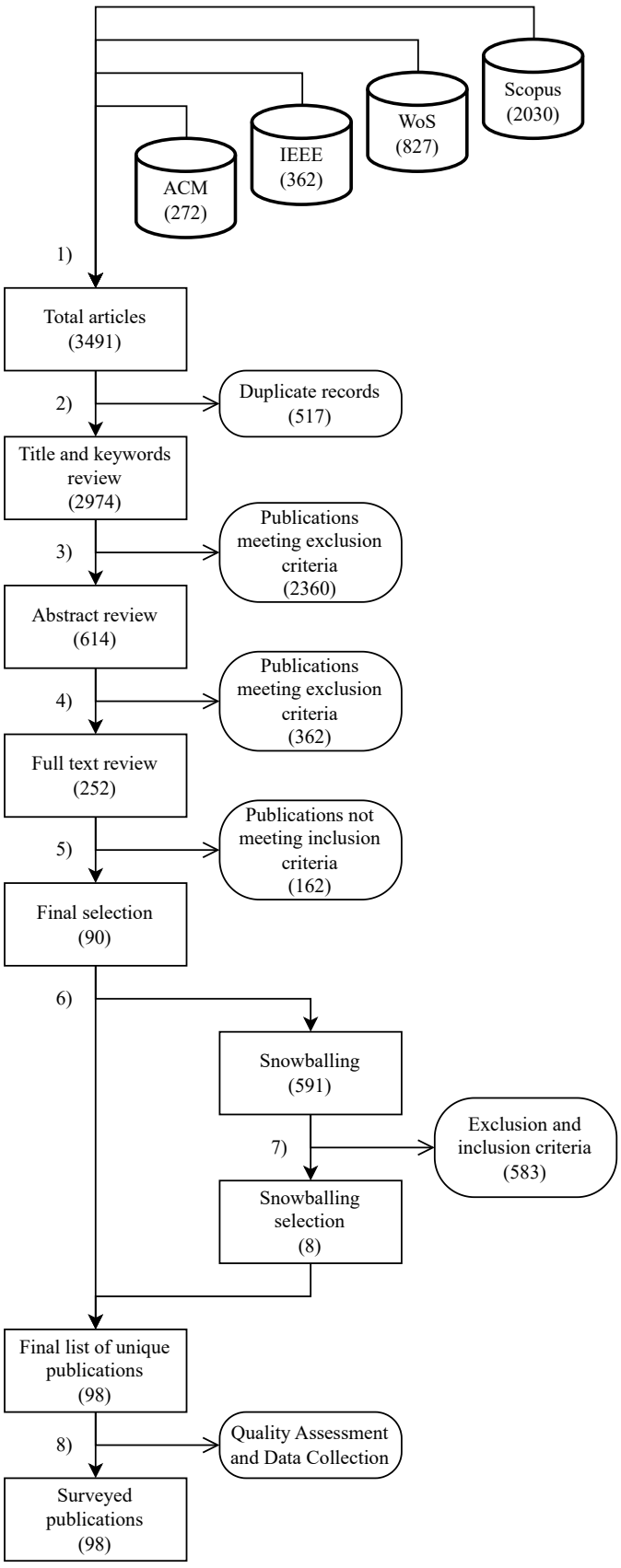

Figure 1: Overview of the search process

7. We filtered the 591 studies in the initial snowballing list by applying exclusion and inclusion criteria to their title and keywords, abstract, and full text, with eight studies meeting all of the inclusion criteria and not meeting any exclusion criterion. We added these eight studies to the previous final selection.

8. With the final list of 98 unique publications, we performed a quality assessment and data collection of every primary study in the final selection to collect information that was suitable to answer our research questions.

As a result of this process, we confirmed all 98 selected studies as final sources. It is relevant to note that we could not retrieve three full-text study reports from any digital library. We contacted the authors, but only one of them answered. Thus there were two studies out of the 252 in the full-text review that we could not review.

The following subsections explain the inclusion/exclusion criteria, the quality assessment process, and the data extraction process.

3.2.4. Selection criteria

To answer our research questions, we evaluated the retrieved studies according to the following exclusion and inclusion criteria. All of the studies in our review had to be relevant to Software Engineering (classified under a SE topic) and focus on computer games. We discarded all of the studies that met any exclusion criteria from our study. In contrast, the studies that met all of the inclusion criteria were the ones selected for our study. Our process had six stages of filtering using the exclusion or inclusion criteria: identifying all relevant studies (3,491), removing duplicate records (517), excluding publications by title (2,974), excluding publications by abstract (362), excluding publications by full text (162), and excluding publications from the snowballing list (582).

The following criteria state when we excluded a study from our study:

- Studies that were duplicates of other studies.
- Studies that were not written in English.
- Studies that were not published online between 2009 to 2021.
- Studies presenting non-peer-reviewed material.
- Studies presenting peer-reviewed but not published in journals, conferences, or workshops.
- Studies that were summaries of conferences/editorials.
- Non-primary studies.

The following criteria state when we included a study in the study:

- Studies that were not focused on the social and educational impact of video games, such as serious games.

- Studies that were not focused on Artificial Intelligence (AI).
- Studies that were not focused on Content Creation.
- Studies that were in the field of Software Engineering.
- Studies that were focused on software engineering applied to industry-scale computer games development.

During this process, we found several interesting studies which we decided to exclude for not being closely related to computer games or software engineering issues. The main focus of these studies is on "artificial intelligence", "mobile gaming", virtual reality", "augmented reality", "content creation", and "serious games". The term *serious game*, as explained by Cheng et al. [48], *"has become a popular and particular term referring to any kind of video game-based learning and training (e.g., business, military, medical, marketing) or the so-called 'edutainment'"*. We did not include studies about serious games because these studies focus on the health, social, or educational implications of games and not on the software that runs them or the engineering methodologies used to develop them. Similarly, studies on "content creation" and "artificial intelligence" for video games are more focused on the content itself and applications rather than the SE behind them [49]. It is also important to note that we only considered full primary studies from journals, conferences, and workshops for the review. We have removed other studies, such as posters, summaries, and non-peer-reviewed studies, from the survey.

Since the selection process is not affected by the order of execution of the selection criteria, we present the criteria in no particular order. Fig. 1 shows the entire process that we followed with the number of excluded studies and included studies.

The final selection consisted of ninety-eight (98) studies. These studies are presented in the *Studies Included in the Review* section at the end of the study report along with the complete data-set extracted from the study in the APPENDIX.

3.3. Quality Assessment

The assessment of the quality of primary studies is critical in a SLR [34] since it helps validate that the methodology and results of these studies are solid. Similarly to Ampatzoglou et al. [5], the quality of the published studies is satisfactory because of the high standards of the publishing process of the selected journals, conferences, and workshops. That is why our quality criteria were not used for filtering purposes to include or exclude primary studies in the review. As Kitchenham et al. [50] proposed, we assessed primary studies using questions. We show these questions in TABLE 2. Similar to Galster et al. [51] and Ali et al. [52], we adopted the quality assessment instrument used by Dyba et al. [53], and we decided to use a three-point scale to answer each question, either as *"Yes"*, *"To Some Extent"*, or *"No"*. Using a three-point scale, we avoided neglecting statements where authors provided only limited information to answer the assessment questions [52]. We assigned each quality assessment question a numerical value as answer (*"Yes"* = 1, *"To Some Extent"* = 0.5, and *"No"* = 0). A quality score was given to a study by summing up the scores for all of the questions asked [52]. We provide the results of the quality assessments in Fig. 7. However, as TABLE 2 shows, these scores refer to the quality of the reporting of a study rather than its actual quality [51]. As we explained in Section 3.1, by selecting only studies with industrial relations, we ensure that the actual quality of the results matches the high standards of the industry.

Because our questions for this quality assessment (shown in TABLE 2) are the same as the ones used in [51], we also followed their approach by applying the quality criteria to all of the studies included in the review rather than only to studies that report empirical methods.

Table 2: Questions to Assess Study Quality

N	Question
Q1	Is there a rationale provided for why the study was undertaken?
Q2	Is there an adequate description of the context (e.g., industry, laboratory setting, products used, etc.) in which the research was carried out?
Q3	Is there a justification and description for the research design?
Q4	Is there a clear statement of the findings, including data that supports findings?
Q5	Did the researcher(s) critically examine their own role, potential bias, and influence during the study?
Q6	Are limitations and credibility of the study discussed explicitly?

3.4. Data Extraction

We show the information collected from the studies in TABLE 3. This data is tabulated and statistically analyzed to investigate the intensity changes of the research activity on the topic, the most active journals and conferences in GSE research, the SE topics currently addressed in GSE research and the number of studies employing each research approach and method. We collected the data for F1 to F7 from the meta-information of study reports. We collected F1, F3, F6, and F7 to keep that information of the study reports. F2 records the year of publication and helps to answer all of our RQs as they are time-related. F4 and F5 help manifest the most active research publication outlets and publishers for GSE studies. F8 records the quality score for each study. F9 records software engineering research topics based on the ACM classifications of 1998 and 2012. F10 and F11 record each study's research approach and method based on the selection criteria introduced in Section 3.1. F12 records the limitations or threats addressed by the authors of each study.

Table 3: Data collection form

N	Field	Research question
F1	Author(s)	n/a
F2	Year	RQ1, RQ2, RQ3, RQ4
F3	Title	n/a
F4	Venue (journal or conference)	RQ1
F5	Publisher	RQ1
F6	Keywords	n/a
F7	Abstract	n/a
F8	Quality score	RQ5
F9	SE research topic	RQ2
F10	Research approach	RQ3
F11	Empirical research method	RQ4
F12	Limitations	RQ5

3.4.1. Classifying topics

This section clarifies the classification of each study concerning the software engineering issue it involves. The classification system employed is the ACM Computing Classification System (CCS). In 2012, ACM undertook a major revision of the CCS, but we classified the primary studies using the 1998 version in order to compare our study with Ampatzoglou and Stamelos' study [5]. We also included the 2012 version classification for two main reasons: to check how appropriate it is for GSE classification and to be able to compare future work to ours by using the newest version of this classification system if needed. The topics used for the 1998 ACM CCS version are shown in TABLE 4, and the topics used for the 2012 ACM CCS version are shown in TABLE 5.

The topics shown for either classification system are not the complete list of categories. Each one of the topics encompasses more subtopics. We used the classification systems with that depth of detail in order to ensure that we could group the results into representative topics.

Both reviewers performed the primary study classification for both CCS versions while performing the full-text review. For the classification process, the authors checked if the study under review was already classified under one of the two versions of the CCS. We reviewed studies dating from 2009 to 2021; therefore, some of them still used the 1998 version instead of the 2012 version. If the study was already classified with either of the two CCS versions, the author currently reviewing the study maintained that topic and classified it for the other version of the CCS. The author did this for the two versions if it was not classified using any CCS version. When the two authors reviewed and classified the study.

It is also worth noting that all software testing and usability testing are included in the topic *5.3.4 Software verification and validation* for the 2012 version of the CCS since they did not match well in any other topic.

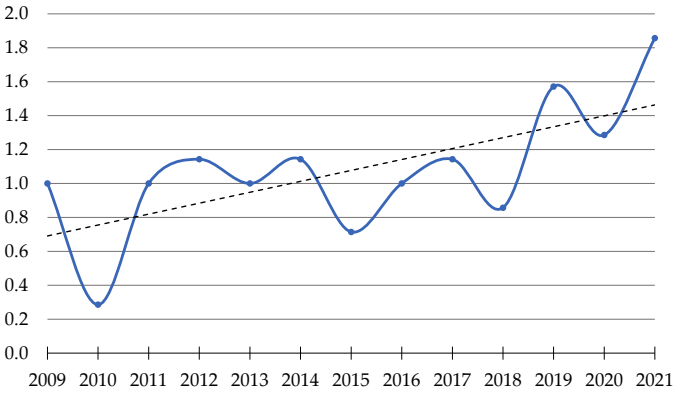

Figure 2: Increase in GSE Research Activity

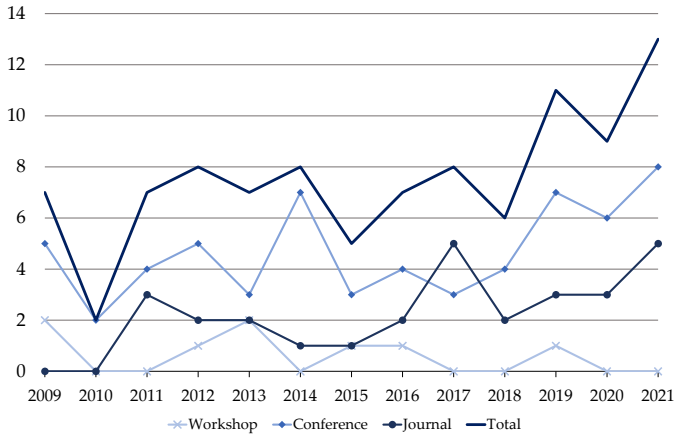

Figure 3: Research activity per year and publication type

4. Results

This section shows the data extracted from the Literature Review of the 98 studies concerning the research questions specified in Section 3.1. Additionally, the data is compared with the research activity of the previous decades. To compare with the data until 2009, we consider the study by Ampatzoglou and Stamelos [5], but our selection criteria are more restrictive because our study focuses on industry-scale computer games. In order to address this issue, we have reviewed how many studies of the final selection in Ampatzoglou and Stamelos' study [5] match our criteria. The previous SLR identified 84 studies from 1994 until 2009 (15 years). Of those studies, only 24 are industry-scale. We show these studies in the APPENDIX. The number indicates the position on the list of studies provided by their authors. Our SLR found 98 industry-scale studies from 2009 to 2021 (12 years). The data-set of the whole search process, selection process, data collection, and quality criteria are available at https://svit.usj.es/SLR-GSE-dataset/ and at https://doi.org/10.5281/zenodo.8242657.

4.1. RQ1 Research Intensity

The most straightforward comparison to address the evolution of the research intensity is counting the final number of studies on this topic before and after 2009. There are 24 studies before 2009 to compare to our data-set of 98 studies after

Table 4: GSE Research Topics CCS '98

Software Engineering Topic	Before 2009		After 2009	
D.2.0 General	2	8.33%	2	2.02%
D.2.1 Requirements/Specifications	9	37.50%	8	8.08%
D.2.2 Design Tools and Techniques	0	0.00%	15	15.15%
D.2.3 Coding Tools and Techniques	3	12.50%	6	6.06%
D.2.4 Software/Program Verification	0	0.00%	6	6.06%
D.2.5 Testing and Debugging	3	12.50%	12	12.12%
D.2.6 Programming Environments	2	8.33%	3	3.03%
D.2.7 Distribution, Maintenance and Enhancement	0	0.00%	3	3.03%
D.2.8 Metrics	0	0.00%	10	10.10%
D.2.9 Management	5	20.83%	12	12.24%
D.2.10 Design	0	0.00%	5	5.05%
D.2.11 Software Architecture	0	0.00%	10	10.10%
D.2.12 Interoperability	0	0.00%	2	2.02%
D.2.13 Software Reuse	0	0.00%	4	4.04%

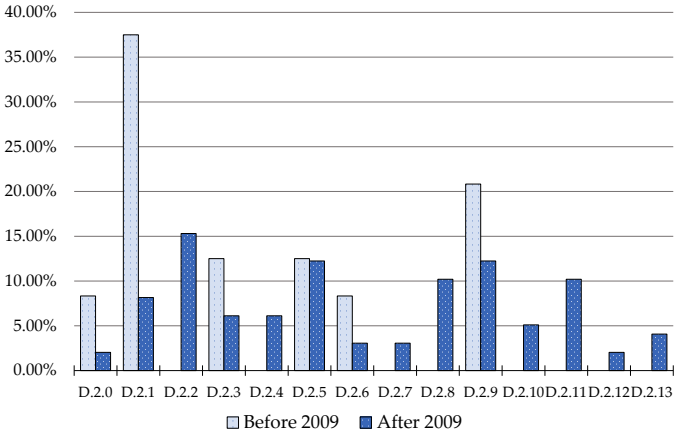

Figure 4: Research Topics

2009. There has been over four times more industry-scale research from 2009 to 2021 than before 2009. Furthermore, the time interval in which we conducted our study is three years shorter.

Fig. 2 shows an increase in the research intensity within our time frame. It shows the number of publications relative to the first year of the time frame, 2009. In addition to the increase compared to previous years, GSE research has kept increasing until now and shows a tendency to keep growing in the future.

Figure 3 shows the distribution of publications per year divided by type of publication: Journal (29), Conference (61), or Workshop (8). Most of the studies are published in conferences, followed by journals, and lastly, workshops. We found 86 different venues of publication for the 98 studies.

4.2. RQ2 Research Topics

TABLE 4 and Fig. 4 show the comparison between the topics of our data-set and the previous data-set [5] filtered by our selection criteria excluding all non industy-scale. It is worth mentioning that every single topic has been researched more after 2009, with the exceptions of *D.2.0 General*, with the same number of studies found (two), and *D.2.1 Requirements/Specifications* with only one study less (from nine to eight).

With the increase in research intensity and maturity, the field has also evolved to be more evenly distributed among the different research topics, as shown in TABLE 4 and Fig. 4. The main topic before 2009 was *D.2.1 Requirements/Specification*. After 2009, we find studies on every topic categorized in the 1998 version of the ACM CCS, emphasizing topics *D.2.2 Design Tools and Techniques*, *D.2.8 Metrics*, and *D.2.11 Software Architecture*. These are the three topics with the biggest increase in interest by the community and part of the five topics most studied, along with *D.2.5 Testing and Debugging* and *D.2.9 Management* after 2009.

TABLE 5 shows how the 2012 version of the ACM CCS topics does not represent GSE as its previous version. In contrast

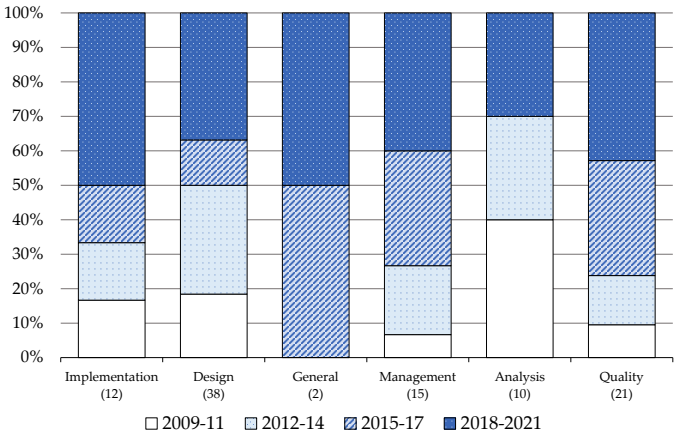

Figure 5: Research Topics Growth Timeline

Table 5: GSE Research Topics CCS '12

Software Engineering Topic	#	%
5.1.1 Contextual software domains	4	4.04%
5.1.2 Software system structures	14	14.14%
5.1.3 Software functional properties	2	2.02%
5.1.4 Extra-functional properties	3	3.03%
5.2.1 General programming languages	0	0.00%
5.2.2 Formal language definitions	1	1.01%
5.2.3 Compilers	0	0.00%
5.2.4 Context specific languages	3	3.03%
5.2.5 System description languages	2	2.02%
5.2.6 Development frameworks and environments	3	3.03%
5.2.7 Software configuration management and version control systems	0	0.00%
5.2.8 Software libraries and repositories	0	0.00%
5.2.9 Software maintenance tools	0	0.00%
5.3.1 Designing software	14	14.14%
5.3.2 Software development process management	13	13.27%
5.3.3 Software development techniques	9	9.09%
5.3.4 Software verification and validation	26	26.26%
5.3.5 Software post-development issues	4	4.04%
5.3.6 Collaboration in software development	0	0.00%
5.3.7 Search-based software engineering	0	0.00%

to its previous version, the 2012 one shows the studies as unevenly distributed, with empty topics and a small number of topics that are very populated.

Fig. 5 shows the timeline of publications per research topic. In order to make this chart more legible, we have grouped the years into four sets of three years except for the last one, which contains four years. For the same reason, we collapsed the fourteen CCS '98 topics into six broader topics, as Ampatzoglou and Stamelos did [5].

It can be observed that more than 70% of the research has been done in the second half of the time frame for *General*, *Management*, and *Quality*. *Implementation* and *Design* have continued to grow in the last few years with more than 65% and 50% publications, respectively. The only exception is *Analysis*, which presents more intensity in the first half of the time frame.

The rest of this section features the description of primary studies to represent the state of the art of GSE. We grouped these descriptions by the Research Topics CCS '98. Each description has a citation to the list of studies included in this SLR. It is relevant to note that we separated the list of the review studies from the list of references, so the citation to the review studies can be distinguished by the *S-* prefix. Additionally, the reader can find a table filled with the full study data-set in the APPENDIX.

4.2.0. General

In two studies, [S1, S2], the authors test the security of industry-scale online video games.

4.2.1. Requirements/Specifications

In two studies [S3, S4], the authors propose that semantic models can be beneficial for the design and development of video games. In [S5, S6], the authors explore the definition of video games as multi-agent systems. In [S7], the authors explore how requirements engineering is used in the video game industry.

In [S8], the authors perform a qualitative study regarding the tool infrastructure of game industry developers.

Two studies [S9, S10] interview game developers to illustrate the experiences of trying to make accessible games and how they manage requirements engineering, respectively.

4.2.2. Design Tools and Techniques

Two studies highlight the importance of analyzing the experience that players have with video games in a pragmatic and hedonic way (from a cultural and emotional perspective) [S11, S12].

In two studies [S13, S14], the authors focus on accessibility by proposing eye-gaze interaction systems. Two studies [S15, S16] focus on improving the User Interface (UI) and reviewing its relationship with the game world. Another study [S17] interviews game developers to illustrate the most widely used practices for achieving an optimal user experience. In [S18], the authors presented a usability inspection method that can be used in the early stages of Model-Driven video game development.

Furthermore, two studies propose the usage of Model-Driven Development in video games, comparing it to Code-centric Development [S19] or using it with Software Product Lines (SPL) [S20]. In four studies [S21, S22, S23, S24], the authors investigate modelling in video games, focusing on the design and its analysis. Finally, in [S25], the authors examine the potential of using Text-To-Speech (TTS) to prototype voice acting in game development.

4.2.3. Coding Tools and Techniques

In [S26], the authors focus on the video game domain to explain a technique of parallel software that is focused on data movement. Four studies deal with rendering [S27, S28, S29, S30], explaining illumination algorithms and 3D rendering effects used in the industry. These provide a technique for increasing modularity and encapsulation of shaders in real-time graphics APIs, and thus explore which rendering technique provides the most attractive results for the users.

Additionally, in [S31], the authors propose a framework and methodology for optimising the building of systems, for example this framework can be used for building shader systems.

4.2.4. Software/Program Verification

Two studies deal with mathematical models for verifying and validating video game gameplay and logic [S32, S33]. In three studies, the authors create frameworks for analysis and evaluation, focusing on Real-time Strategy (RTS) games [S34], the usability of video games developed with Model-Driven Development [S35], and Unreal Engine 4 blueprints [S36]. In [S37], the authors detect hot spots in the code of video games.

4.2.5. Testing and Debugging

Two studies explore new testing methods, where the authors present an incremental method for extracting adaptive tests from formal, object-oriented specifications [S38] and use combinatorial testing to establish a framework to improve the testing efficiency of video games [S39]. Another two studies describe approaches for automatic verification and testing in node-based visual script notation [S40] and code [S41].

Furthermore, one of the most important tasks of video games is to provide quality entertainment to their players, so they need a significant amount of testing on user-centered issues, usually through playtesting. In two studies, experiences and methods of playtesting are described and performed for independent video game development studios (*indie* studios) [S42, S43]. The second study also explores the differences between the highlighted features in game review analysis and in play-test reporting. In [S44], the authors explore the usage of players' game reviews as a cheap and effective way of collecting user feedback.

Moreover, crashes in video games significantly produce a negative user experience, so a new crash reporting system is proposed in [S45], improving data reliability, data availability, and query responsiveness while reducing CPU and memory usage. In [S46], the authors investigate how to collect and centralize events with a distributed system for debugging in online video games. In addition, in [S47], the authors propose a taxonomy for bug types after analyzing common existing bugs in industrial games, and, in [S48], the authors focus on fixing bugs at runtime.

Finally, in [S49], the authors study the differences and similarities between software and game industries on testing and quality assurance.

4.2.6. Programming Environments

In [S50], the authors present an engine is presented focused on creating visual novels. Another study compares open source game engines with traditional software frameworks finding differences [S51]. Finally, [S52] presents a tool to ease communication among game development departments.

4.2.7. Distribution, Maintenance and Enhancement

In [S53], the authors explore how Lehman's laws of software evolution are applied in video games. In two more studies [S54, S55], the differences between traditional software and video games patches are highlighted, providing an extensive analysis of the impact on the users after every patch in an industry-scale competitive online video game and how these affect players' engagement.

4.2.8. Metrics

In four studies [S56, S57, S58, S59], the authors investigate the capture of metrics. Furthermore, three studies [S60, S61, S62] present how to analyze metrics and telemetry data based on the player's gameplay for the improvement of the next video game titles or pattern generation. Finally, in [S63, S64, S65], the authors present the definition and usage of new and adapted usability heuristics.

4.2.9. Management

In [S66], the authors introduce a maturity model that focuses on evaluating video game development methodologies in an organization. Also, in [S67], the authors attempt to adapt the ISO/IEC 29110 to be applicable to the video game industry, concluding that very small companies could benefit from it. Furthermore, some studies investigate how traditional Software Engineering management techniques could be applied to video game development [S68, S69] or compare them with techniques and frameworks that are already being used in video game development [S70]. Some studies explore how development life cycle models could be used in video games [S71], what framework video game development processes should follow [S72], and how to manage and improve communication among the members of a development team [S73].

In [S74], the authors present a survey exploring risk management in the video games industry. In [S75], the authors also present a survey along with postmortem examples to study how crunch affects the industry and its employees. In one study [S76], the authors present a survey to professional game developers about technical debt, a problem that they are aware of for which almost no solution has been implemented. Finally, [S77] presents a survey dealing with how video game companies are using SCRUM sprints.

4.2.10. Design

Three studies concern the Game Design Document (GDD), reviewing how GDDs are written in the industry with no standard and proposing new guides for building them [S78, S79, S80]. In [S81], the authors use a gameplay data analysis to fix and improve game design. In [S82], modding is described as a common practice among video game communities where open-source video game software is customized.

4.2.11. Software Architecture

With regard to the network architecture in online games, in [S83], the authors propose a software architecture for traffic generation in massively multiplayer online role-playing games (MMORPGs), whereas in [S84] the authors analyze networking approaches of online video games and exploitable security concerns. In [S85], the authors introduce a high-performance and robust rendering pipeline. In [S86], a survey deals with the differences in the usage of architectures between traditional software and video game development and its evolution in the last years.

In addition, in two studies [S87, S88], the authors present an evaluation of a simple method to find architectural problems in a product line of computer games that use the Model-View-Controller (MVC) pattern, and they apply the MVC pattern in video game development to prove that it solves common problems. [S89] employs multi-threading in a video game engine to demonstrate that the performance is improved. In [S90], the authors describe an architecture for economy systems in video games that reduces their implementation and maintenance costs. In [S91], the authors theorize about formal models being useful for video game development. In [S92], the authors

explore how the multiple platforms in the market affect both software and video game development.

4.2.12. Interoperability

In two studies [S93, S94], the authors explain how they created middleware for multiplayer games.

4.2.13. Software Reuse

In [S95] presents an engine allowing generic platform-independent video game creation. Two studies [S96, S97] deal with DSLs for code generation at runtime for the video game domain.

Finally, in [S98], the authors focus on code transpiling to solve the issue of changing a game engine that is currently in use while in development.

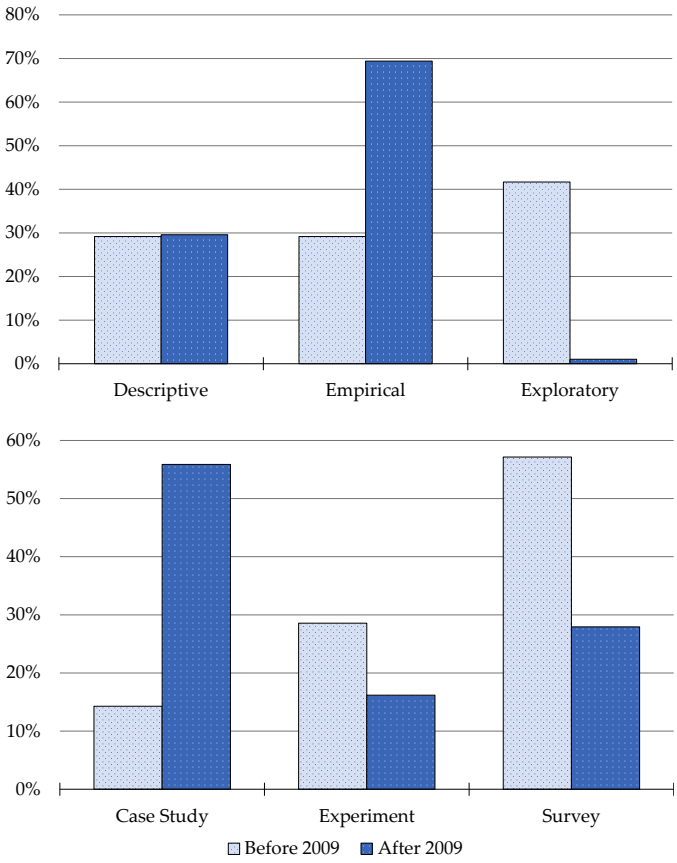

Figure 6: Research Approaches and Empirical Research Methods

4.3. RQ3 Research Approaches

GSE practitioners have shifted their research approach drastically in the last few years, as shown in Fig. 6. Currently, there is almost a complete lack of Exploratory studies, decreasing from more than 40% to 1% of the studies. In contrast, empirical methods have more than doubled in interest in the last few years. It is worth noting that, in the last decade, exploratory papers are almost non-existent for industry-scale computer games.

Figure 7: Publication count per quality total

4.4. RQ4 Empirical Methods

According to Fig. 6, the case study was the least widely conducted empirical method by researchers before 2009, but it has evolved to be the most widely conducted method since 2009 in more than 50% of the empirical studies.

4.5. RQ5 Research Quality

With regard to the quality questions elucidated in Section 3.3, we find that most studies got *"Yes"* or *"To Some Extent"* to the questions about motivation and context (Q1 & Q2). Fig. 8 also reveals how the justification of the research design and the inclusion of data that support findings (Q3 & Q4) is less addressed by the community. Nevertheless, more than 70% of the studies tackle them, at least, to some extent. Finally, GSE researchers generally lack an explanation of the critical examination of their own work, either of potential biases and influences affecting their study or the limitations of that work (Q5 & Q6).

There are 41 out of 98 studies that address (at least to some extent) the limitations of the study, and 13 out of 98 address their own potential bias or influence. The questions least addressed by the community (Q5 & Q6) generally comprise the section devoted to study limitations, threats to validity, or similar. This section only appears in 13 studies. The rest of the studies that acknowledge their study validity do it briefly inside other sections related to method, discussion, or conclusion. Even when the studies present threats to validity, 13 do not present mitigation strategies, another 13 studies perform the mitigation, and 15 intend to pursue it as future work.

It is worth noting that only 7 studies use best practices or widely accepted methods. The validation of other studies relies on performance evaluations or user acceptance by conducting questionnaires to practitioners in the industry.

5. Discussion

5.1. RQ1 Research Intensity

As games grow in size and complexity, developers need new techniques and approaches to manage their ambitious software. We observed in Section 4.1 that the number of studies on GSE has been growing over time and that interest in GSE research matches the continuous growth of the video game industry [1]. The GSE research community is responding by bringing innovation to them.

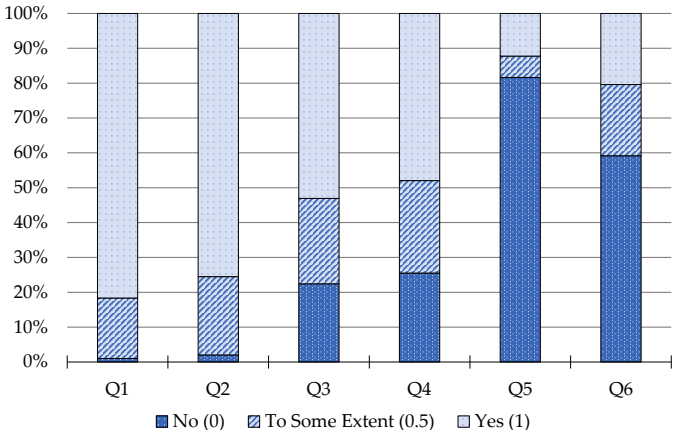

Figure 8: Research Quality

The 98 studies are dispersed among 86 different venues of publication. These results show that, while there are emerging venues out there that are focused on GSE research (such as IEEE Transactions on Games, Conference on Games, ICSE Workshop on Games and Software Engineering, or Workshop on Automated Software Engineering for Computer Games), the research community in this field is still scattered. The full list of publications by venue is displayed in the supplementary material.

It is worth noting that, even though there are no focused venues, these are linked to a few publishers. IEEE holds the majority of publications, with 40% of the publications. This is in contrast with the previous years, when ACM was the main publisher [5].

The growing body of research dedicated to Game Software Engineering (GSE) signifies a rising interest in enhancing the development and maintenance of video games. This surge in scholarly attention highlights the recognition of games as complex software systems that require specialized engineering practices to ensure their quality, stability, and efficient production. As the video game industry continues to expand, the significance of GSE research becomes even more apparent, showcasing a direct response to the industry's demand for more effective tools and techniques.

5.2. RQ2 Research Topics

Before 2009, the main topic in GSE research was *D.2.1 Requirements/Specification*. It is usual for a young domain to attract researchers on this topic [5].

Video games are a special type of software with a major focus on user experience and interface, as video games require the player to perform a great range of actions while at the same time being enjoyable. At the same time, they are complex systems with many subsystems of high quality and a large amount of content that interact with each other. Some studies on *D.2.2 Design Tools and Techniques* focus on user experience and interface, while some of them focus on the documentation of the systems. Studies on *D.2.11 Software Architecture* focus on the architecture of those systems. These two topics tackle the challenges of size and complexity that characterize the high-standard industry of video games. This is probably a reason behind the increase in research on those two topics.

The increase of interest in *D.2.8 Metrics* can also be explained by the nature of this domain of software. These large systems, along with the high population of players, can provide plenty of data, which helps improve game experience and performance. This leads to researchers and industry developers growing interested in getting and managing heuristics from the software usage.

It is also worth noting that the interest in *D.2.5 Testing and Debugging* is similar before and after 2009. Following the previous reasoning, video games as software are big, with a high density of features that can increase unwanted behavior. In this field, there is also a difference between *software testing*, which focuses on errors in the logic of the systems, and *game-play testing*, which considers meeting the standards of quality and the player's experience. These standards deal with the correct enjoyment, quality, and performance. The increase in interest in topic D.2.5 goes together with the high amount of effort and resources that the industry puts into testing [5].

We observed that *D.2.9 Management* is the second most studied topic. As seen in Section 2, development teams are bigger and more heterogeneous than traditional SE teams. The fact that practitioners and researchers are focusing on this topic means that managing large software and teams is still an ongoing problem that needs to be addressed because of the great rise in super productions of video games.

The studies in the 2012 version of the ACM CCS topics are very unevenly distributed, with empty topics and a small number of topics that are very populated. The best example is the topic *5.3.4 Software verification and validation*. Most of the studies that are categorized in three different topics from the 1998 CCS (*D.2.1 Requirements/Specification*, *D.2.5 Testing and Debugging*, and *D.2.8 Metrics*) are all collapsed inside the 5.3.4 topic from the 2012 CCS. Also, these three categories have proven to be of great interest to the community. The 2012 ACM CCS replaces the traditional 1998 version of the ACM CSS in serving as the de facto standard classification system for the computing field [39]. However, it does not render the current state of the GSE research domain correctly, making the differences between these two fields more evident.

We observed that research related to *Analysis* has more intensity from 2009 to 2014, while more than the 70% of the research related to *General*, *Management*, and *Quality* has been done from 2015 to 2021 and the intensity of the research related to *Implementation* and *Design* has continued to grow from 2018 to 2021. The video game industry experienced significant growth during the early 2010s, with the rise of mobile gaming [54] and indie game development [55]. This expansion may have led to an increasing volume of research on management for new emerging development teams of different sizes and on quality to meet the demands of a growing and diverse player base. In later years, as the market became more competitive, there could have been a greater emphasis on design and implementation aspects to stand out from the competition by offering a software aspect that demonstrates superior performance and scalability.

Finally, we observed some established and emerging trends in the reviewed studies, which are discussed further in Section 6:

- Model Driven Engineering appears in multiple studies on different aspects of GSE [S18, S19, S22, S35, S36].
- Software reuse appears to be a recurring practice and subject of research for both different products and different platforms [S95, S50, S98, S89, S92, S8, S20, S36, S92, S1].
- The approach reported in many studies is evaluated in a specific environment and it is not trivial to generalize it [S55, S54, S34, S62, S2, S13].
- The implementation of traditional software engineering methodologies in game development teams often proves ineffective [S66, S70, S71, S77, S75].
- Games as a Service (GaaS) are becoming increasingly prevalent in the gaming industry [S54, S55, S62, S2, S94, S93].
- Multiplayer online games create the demand for reliable, robust, and secure systems that operate without compromising performance [S93, S94, S83, S2].
- Many studies propose the use and improvement of Game Design Documents (GDDs) for better communication and documentation [S4, S5, S6, S78, S79, S80, S77]. However, an established standard for them is yet to be achieved and they do not seem to be a definitive solution.
- Among the requirements that need to be defined in video games, there are several differences compared to traditional software, and these requirements have not yet been addressed specifically [S7, S3, S4, S5, S6].
- Testing is another recurring topic in GSE since it is more challenging to automate than traditional software [S38, S39, S40, S41, S42, S43, S44, S46, S48, S49].

As the gaming landscape evolves, games are expanding in size and intricacy, necessitating new strategies and methodologies to manage the complexity that arises from their development effectively. Developers are confronted with the challenge of creating games that offer expansive worlds, interconnected mechanics, and rich narratives, which in turn demand innovative approaches to ensure smooth software development. This drive for innovation stems from the recognition that traditional software engineering practices might need to address the unique challenges presented by game development.

5.3. RQ3 Research Approaches

Exploratory studies are usually expected to be more prominent in a young research area and interest in this research approach decreases as the field develops [5]. This may be why it is the least exciting approach for practitioners. There is almost a complete lack of studies using this approach, decreasing from more than 40% to 1% in the last decade. This decrease may be a cause for the lack of quality in the report about the studies. We addressed this in Section 4.1, where the issues related to the validity of the research are poorly questioned.

Empirical methods have more than doubled in interest in the last few years. As GSE has grown in maturity, companies have noticed the benefits of collaborating with researchers and have started doing it in a more active way. This has increased the number of empirical studies and decreased the gap between industry and academic research. This results in practitioners presenting industry-scale demonstrations for their studies. The hypothesis of GSE growth in maturity is also supported by the results, which conclude that the field has also evolved to be more evenly distributed among the different research topics.

Along with the increase in intensity, the concurrent trajectory of GSE research methods underscores a meaningful alignment between research and industry. This implies that researchers are proactive in addressing the evolving challenges stemming from developing sophisticated games. This alignment fosters a symbiotic relationship between academia and industry, as research innovations are poised to directly contribute to improving game development processes.

5.4. RQ4 Empirical Methods

The fact that case studies are the empirical method that has been most widely conducted by researchers since 2009. This can be a sign that the gap between industry and academic research is smaller since surveys were previously the majority, and now case studies are. In the past, companies were only involved in research at a surface level, answering surveys. Now there are plenty of case studies that nourish GSE.

Experiments were also expected to increase since 2009 [5], but the research evidence does not support this hypothesis. The video game industry is fast-paced, having less development time than traditional software [56]. This rush can explain the reduced number of experiments applied. Experiments require more effort and are more time-consuming than case studies, which have a strong short-term impact with direct implementation of the product.

GSE research makes use of in-depth case studies that are drawn from industrial game development initiatives. This collaborative synergy not only strengthens the relevance of GSE research but also leads to the creation of practical tools and techniques that address the specific needs of the gaming industry. As companies openly share their experiences and challenges, the research community gains access to invaluable data and context, ultimately reducing the gap between theoretical advancements and pragmatic modern game development.

5.5. RQ5 Research Quality

In Section 4.5, we stated that best practices or widely accepted methods to report limitations are only used by seven studies. This situation can be explained by the lack of exploration in GSE research strategies and practices due to the empirical nature and relative youth of this field.

The average score of all the studies is 3.49, with a standard deviation of 1.34. Although the field of research has matured to

the point where there are a lot of industry-scale demonstrations, the study reporting needs to be more rigorous.

It becomes imperative to ensure that research outcomes are communicated with a higher degree of clarity, transparency, and repeatability. Researchers must acknowledge the importance of adopting standardized methodologies, rigorous experimental designs, and systematic documentation practices to effectively validate and communicate their findings. This will lead to meaningful and informed decision-making in both academia and industry.

6. Future Directions

In this section, we analyze the different research challenges that are open in the field of work of GSE. We present in the following subsections the challenges, why they exist, and the open problems for each challenge. We also provide recommendations and potential future research lines in the identified areas.

6.1. Cross-fertilization between GSE and MDE

GSE is a fertile field for Model Driven Engineering (MDE) research to thrive [S18, S22, S19, S36, S35]. The heterogeneity of the profiles involved in the creation of a video game makes it necessary to have approaches that allow profiles without requiring specific programming knowledge to participate in the creation process. In engines such as *Unity* and *Unreal Engine*, where many development processes have been presented in an abstract way through models, templates, blueprints, and flowcharts, reducing the prominence of the source code generated from these models. This practice is what the MDE research community has been pursuing for decades in classic software engineering. Nevertheless, the MDE research community does not seem to be fully aware of the modeling success achieved by video games given the tiny presence of GSE work in MDE venues. There is an opportunity for the MDE research community to learn from the success of models in video games in the hope of transferring it to other SE domains. It also provides the GSE community with an opportunity to leverage MDE approaches for video game models.

6.2. Game Software Product Lines

GSE is a domain where software reuse is constant, either adapting systems and elements from different games in a company or within the same game (the same element can appear in different parts of the same game) [S98, S89, S36, S20, S92, S8]. The urgency of the sector to meet deadlines hinders the investment of time in developing and implementing specific techniques for code reuse. Some advances have been made towards this, such as the development of game engines [S95, S50]. However, other advances in reuse have not yet been applied with the same success as in the case of software product lines and the planned reuse approach they propose. We think that in the near future there will be a growing body of work from the GSE reuse community that leverages SE reuse techniques to the GSE world and adapts them for maximum benefit. Software reuse is also a key feature in porting video games to different platforms.

6.3. The Blossoming of Publishing Platforms

There are many platforms on which to publish video games. With the growth of the video game industry, multi-platform publishing is the norm. These platforms have different input schemes, different computing power, different audiences, different operating systems, and different hardware [S98, S92, S41, S1]. Video games need to comply with the high industry standards in all of the platforms on which they release their software. These platforms also change rapidly with new console and hardware releases. Porting a game from one platform to another is not a trivial task and developers spend a lot of resources doing it. However, GSE research neglects this topic

Key insights

RQ1: The landscape of industry-scale GSE research is witnessing an x4 growth, with 98 studies published across 86 distinct venues. This surge in activity underscores the intensifying interest in GSE, a field that is evolving but is dispersed.

RQ2: The shift in the topic interest by the researchers indicates an increase in the maturity of industry-scale GSE research. As the video game market becomes more competitive, GSE research provides more sophisticated techniques and methods.

RQ3: Recent years have seen a remarkable doubling in interest in empirical methods, reflecting the maturation of GSE. Industry recognition of collaboration benefits has led to an active increase in empirical studies, narrowing the industry-academia gap.

RQ4: Case studies have surged as the primary empirical method used by researchers since 2009. This shift suggests a further reduction in the industry-academia gap, marking a transition from surveys to more immersive case studies. This change reflects deeper industry involvement in GSE, showcasing a richer understanding of real-world scenarios.

RQ5: While the research field has progressed, marked by a substantial presence of industry-scale demonstrations, there is still a demand for increased rigor in study reporting.

We expect more work on GSE techniques and tools that help developers when porting video games across platforms.

6.4. The Multifaceted Nature of GSE

As previously mentioned in Section 5.2, one challenge that most research in software engineering has (traditional or video games) is the need for generalization of findings. The approach reported in GSE work is usually evaluated in a specific environment that cannot be generalized directly to all cases [S55, S54, S34, S62, S2, S13]. In video games, this problem is amplified due to the inherent heterogeneity of the field. Video games are being made by teams of very different sizes, from solo indie developers to big corporations with hundreds of employees. There are many platforms with plenty of differences among them. There are also plenty of genres that change the software significantly, from physics-based action games in 3D to puzzle games in 2D. The underlying software that composes the genres amplifies the differences in the software between games (and sometimes in the same game), making it difficult to generalize findings for GSE research. There is a need to tackle the challenge of generalization in the context of GSE settings.

6.5. Planning and Scheduling Issues

The video game industry focuses on agile methodologies. Many report that agile is not properly implemented in the teams and there are misconceptions about different agile methodologies [S66, S70, S71, S77]. Crunch time is a term used in the game industry to describe high-stress periods working overtime that can last months when releasing a video game [S75]. The multidisciplinary teams in game development are using agile methodologies as a legacy from traditional software engineering [7], but these methods suffered from crunch time [S75]. The way in which many companies are adopting existing SE methodologies is clearly not the optimal solution when developers need to rely on crunch time. The GSE community can put effort into finding new best practices for creating games and adapting agile methods for game development teams of different sizes in order to avoid crunch culture.

6.6. The Consequences of Games as a Service

The GSE community is expected to follow the growth, maturity, and needs of the industry. The five topics that show the greatest interest from practitioners (D.2.2 Design Tools and Techniques, D.2.5 Testing and Debugging, D.2.8 Metrics, D.2.9 Management, and D.2.11 Software Architectures) will probably hold this research intensity over the next few years as they tackle the problems in an industry that keeps growing in projects and team sizes. Nevertheless, a part of the video game industry has veered towards a business model approach called Games as a Service (GaaS) [57]. Similar to the concept of Software as a Service (SaaS), these are products that do not end development upon release. For this reason, topic D.2.7 Distribution, Maintenance, and Enhancement may see an increase in engagement by both industry and the research community. Because of the GaaS trend, software maintenance will be a major issue in the video game industry in the future [S54, S55, S62, S2, S94, S93].

6.7. GSE Security

Multiplayer online games comprise a significant part of the user base of video games. These games need to build network systems that are reliable, robust, and secure [S93, S94, S83, S2]. Online video games synchronize large amounts of data provided by different platforms without latency in order to be competitive. With the rise of Massive Multiplayer Online (MMO) games, bandwidth usage and high traffic are issues that have greater complexity in the GSE domain. These games portray large complex 3D worlds with physics calculations involved and many players interacting with each other. Online games can also be competitive in nature, giving users a platform for what is called e-sports (tournaments where professional video game players compete). This raises the concern of cheaters and the need for anti-cheat technology that detects players who exploit the vulnerabilities of the software.

6.8. Weaving Software and Art Design

The fact that software, art, design, and sound are tightly connected in video games poses difficulties in documenting the development. Practitioners cannot produce software without taking into account the design that will make the game fun and the art and sound that will surround the experience. The communication among the multidisciplinary team members must be clear and fluid. The GSE research community is already putting effort into this topic by proposing improvements for these documents called Game Design Documents (GDDs) [S4, S5, S6, S78, S79, S80, S77]. Nonetheless, there is no established standard for the making of GDDs; every practitioner develops these documents in their own fashion. To address the evolving needs of game development teams, future directions must focus on improving and exploring methods, possibly veering out of the traditional GDD creation due to the static nature of GDDs that hinders iteration and prototyping [58].

6.9. A New Wave of Requirements Research for GSE

Video game development focuses on the fun and entertainment of the players. Traditional SE usually does not bother with this, making GSE a domain that needs a change in how researchers approach requirements engineering (RE). Even though RE is a topic that has lost attention from the GSE community, research still needs to be done in this field. RE can help define the final user experience desired by developers. It can facilitate tasks of gameplay testing because quality assurance developers will have a clearer objective when testing. We think that, in the future, there will be more RE work applied to GSE that properly exploits the particularities of the domain (e.g., in fields such as user experience evaluation or video game balancing) [S7, S3, S4, S5, S6].

6.10. Testing and Debugging in the Face of Gameplay Subjectivity

Similarly to RE, testing in video games is peculiar due to the player's subjectivity for fun. This divides testing in two when applied to GSE: traditional testing, which searches for software bugs and faults that make the program misbehave [S38, S39, S40, S41, S46, S48]; and gameplay testing, which affects how the user perceives the experience (e.g., the game is unbalanced and unfair) [S42, S43, S44, S49]. GSE has to deal with large code bases with high interactivity among objects and large amounts of assets that make bug reproduction a difficult task that needs to be addressed. Additionally, when performing gameplay testing, the human tester has to spot bugs while playing the game, making this process subjective to human opinion and ideas. There is a need for automation and new techniques in the testing and debugging domain.

6.11. Green GSE

Video games need to comply with high industry standards in terms of performance while creating complex 3D worlds. GSE is a field that needs to take advantage of the hardware acceleration of graphic cards to provide stable execution of the game with constant Frames Per Second (FPS). This makes code performance a significant concern to developers [S86, S89]. It can also affect other aspects of the game and the user experience such as long loading screens, which are states in which the game loads all the needed assets. Additionally, low performance in code leads to high energy usage that affects portable machines such as laptops and mobile phones. Reducing energy usage with better performant software also plays an important role in ecology [59] since millions of players are consuming video games. In the end, having fast code in video games is an open issue that has many implications and the next generation of games will need improvements in code performance.

6.12. Search-Based GSE

Search-Based Software Engineering (SBSE) is a steady and popular topic among the SE community [60]. The key ingredient for SBSE approaches is the fitness function that guides the search. In video games, there are Non-Playable Characters (NPCs), agents with artificial intelligence that can behave like the player. These NPCs can be used to perform simulations, which are an opportunity to guide the search in SBSE, improving the fitness function. Also, the high user base of video games (blockbusters gather hundreds of millions of players) can contribute to the vision of Humans as Fitness Function [61], where humans share the load of assessing candidate solutions.

6.13. Content Creation and Artificial Intelligence

The scope of this review has omitted content creation and artificial intelligence. There is also a need to analyze the state of the art in these domains and how GSE is present in them.

7. Threats to Validity and Quality Assessment

7.1. Threats to validity

Wohlin et al. propose a classification for assessing the validity of the results with four categories of validity concerns [62, 63]. In this section, we test our review against these categories.

Internal Validity: This category can be a validity threat when the results of an investigation may be affected by factors affecting the dependent variables without the researcher's knowledge. In our work, this could be a threat when we selected the studies to review. To reduce the likelihood of erroneous exclusions or inclusions, we defined the criteria for when a study can be excluded or included in Section 3.2.4. We used the same criteria for evaluating all of the studies from the total studies list to the final selection. In the same way, we performed the data collection and the quality assessment of each selected study following the criteria defined in Sections 3.4 and 3.3, respectively.

External Validity: This category addresses the likelihood of generalizing the results of the findings. In order to minimize the threat of the primary studies collected from the databases not being representative of the target population, we used the ACM and IEEE databases in addition to two indexing databases (Scopus and Web of Science), in the way suggested by Petersen et al. [46]. We reported this process in Section 3.2.2.

Conclusion Validity: This category is concerned with the reliability of the relationship between the treatments and the outcome of an experiment. In our case, the results might be compromised if researchers were looking for a specific outcome. To avoid this threat, we conducted the search method using best practices in SLRs [34, 45, 64]. The only case for discarding a study was if it did not match the inclusion criteria or if it matched any exclusion criteria. Also, as explained in Section 3.2.4, two reviewers performed the selection process, and all disagreements were resolved by discussing the decision of each reviewer and consulting a third author if necessary.

Construct Validity: This category is related to the concepts, theories, and measures that the researchers have in mind. *Examples of issues are whether the concepts are defined clearly enough before measurements are defined, and interaction of different treatments when persons are involved in more than one study* [62]. To prevent this threat, we defined the research questions for researchers to know what to investigate, and we defined a search process for all of the reviewers of this work to follow. The research questions are provided in Section 3.1, and the search process is explained in Section 3.2.3.

7.2. Assessment of review

As an assessment of this review, we have performed an evaluation using four quality assessment (QA) questions from the

work of Kitchenham et al. [36] about systematic literature reviews in software engineering. We answer the questions by using the specified criteria by Kitchenham et al.

QA1: *Are the review's inclusion and exclusion criteria described and appropriate?*

Yes, we explicitly defined the inclusion criteria in the study.

QA2: *Is the literature search likely to have covered all relevant studies?*

Yes, the authors have searched four digital libraries with the extra search strategy of snowballing.

QA3: *Did the reviewers assess the quality/validity of the included studies?*

Yes, the authors have explicitly defined quality criteria and extracted them from each primary study.

QA4: *Were the basic data/studies adequately described?*

Yes, we presented information about each study.

These quality assessment criteria might be biased as they are somewhat subjective. In order to minimize erroneous results, two researchers evaluated these quality criteria independently. A different researcher outside of the review process resolved all discrepancies.

7.3. Other threats and limitations

Another threat to the validity of our study is due to the fact that 8.08% of our final selection are workshop reports. The inclusion of these studies in the SLR might have altered the review results due to the nature of these studies with respect to journals and conferences.

Finally, it is worth noting that our review only focuses on computer games and industry-scale studies. Therefore, it might not be safe to generalize it to the whole GSE research. These would also include studies focusing on non-computer games and studies with evidence obtained from academic studies, expert opinions or observations, demonstration or working out academic prototypes, or with no evidence at all.

8. Conclusion

This study presents an SLR to study software engineering research for industry-scale computer games. This review comprises 98 studies portraying the current relationship between GSE academia and industry.

Our findings reveal that research on GSE continues increasing in interest and that the industry benefits from it (with over four times more research than before 2009). Nevertheless, these studies need to be more rigorous with the critical examination of their work.

We have observed an evolution in the research topics that draw the most attention from practitioners. The categories related to a young domain are decreasing, leading to a more mature field with research about architecture and design techniques as the main topics. We expect this trend to continue, with topics related to software maintenance and games as a service (GaaS) gaining relevance in the upcoming years.

The 2012 ACM CCS does not correctly represent the GSE research domain. The studies that we reviewed were unevenly distributed in this categorization. This contrast supports the claim that GSE and traditional SE are diverging. A new classification for GSE might be needed in the near future. In addition, the GSE research community is completely dispersed, with no main venues holding GSE studies yet. We also expect specialized journals and conferences on GSE research to start appearing over the next few years.

The research approaches and empirical methods used by the community support the hypothesis that GSE is an independent, mature, and growing domain. Our study shows how the gap between industry and scientific research is narrowing, even in the rapidly changing industry of video games.

Our results confirm the assumptions from the previous work by Ampatzoglou and Stamelos [5]: *"more elaborate empirical research methods are going to be employed"*, *"a wider range of topics is going to be covered"*, and *"game engineering is a scientific domain rather than a soft skill topic with only a peripheral research activity"*.

This study provides background information for any relevant future GSE work, helping the community to explore the research gaps that GSE presents within an industry that shows great interest in academic research. This indicates a bright future for the growing field of Game Software Engineering. While GSE is still a young field, it is growing and maturing each day. We presented many lines of research for the future, making GSE a fertile domain that is constantly growing.

According to the report provided by S. Shuermans and C. Voskoglou in 2019 [65], there are 18.9M software developers, of which 8.8M are game developers. The research in GSE is relevant to almost half of the software developers in the world and is, directly or indirectly, relevant to all the roles that deal with the other aspects of development: art, music, design, narrative, etc.

The large democratization of game development poses an opportunity for the mainstream engines (Unity and Unreal) to teach and divulge software engineering practices for developers that do not have an engineering background. Developers that built expertise in these engines often lack experience in SE methods and could benefit from them. Not only can large organizations benefit from GSE research, but small video game studios can also cover many roles with a reduced team. In these scenarios, automatizing processes, performing accessible testing practices, or team management methodologies can promote serious changes in development. They can facilitate the technical process and reduce the technical debt of developers so that they can focus on the creative and fun part of the video game craft.

This study provides a bridge between industry and academia so that they can nourish each other. It shows traditional SE researchers that GSE deals with pieces of software that are complex enough to address most topics that traditional SE also cov-

ers. Researchers can find complete and industrial cases for their studies inside video game development. By providing a large sample of research studies in GSE, we hope that industrial practitioners are encouraged to publish their industrial-scale expertise. Video games are powerful tools that can push the boundaries of software knowledge.

Acknowledgements

This work was supported in part by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the Project VARIATIVA under Grant PID2021-128695OB-I00, and in part by the Gobierno de Aragón (Spain) (Research Group S05_20D).

Studies Included in the Review

[S1] L. Chen, N. Shashidhar, D. Rawat, M. Yang, C. Kadlec, Investigating the security and digital forensics of video games and gaming systems: A study of pc games and ps4 console, in: 2016 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2016, pp. 1–5.

[S2] B. D. Bryant, H. Saiedian, A state saturation attack against massively multiplayer online videogames., ICISSP (2021) 217–225.

[S3] J. Kessing, T. Tutenel, R. Bidarra, Designing semantic game worlds, in: Proceedings of The third workshop on Procedural Content Generation in Games, 2012, pp. 1–9.

[S4] C. Marín-Lora, M. Chover, J. M. Sotoca, A game logic specification proposal for 2d video games, in: World Conference on Information Systems and Technologies, Springer, 2020, pp. 494–504.

[S5] C. Marín-Lora, A. Cercós, M. Chover, J. M. Sotoca, A first step to specify arcade games as multi-agent systems, in: World Conference on Information Systems and Technologies, Springer, 2020, pp. 369–379.

[S6] C. Marín-Lora, M. Chover, J. M. Sotoca, A multi-agent specification for the tetris game, in: International Symposium on Distributed Computing and Artificial Intelligence, Springer, 2021, pp. 169–178.

[S7] J. Kasurinen, A. Maglyas, K. Smolander, Is requirements engineering useless in game development?, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2014, pp. 1–16.

[S8] J. Kasurinen, J.-P. Strandén, K. Smolander, What do game developers expect from development and design tools?, in: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, 2013, pp. 36–41.

[S9] J. Kulik, J. Beeston, P. Cairns, Grounded theory of accessible game development, in: The 16th International Conference on the Foundations of Digital Games (FDG) 2021, 2021, pp. 1–9.

[S10] M. Daneva, How practitioners approach gameplay requirements? an exploration into the context of massive multiplayer online role-playing games, in: 2014 IEEE 22nd International Requirements Engineering Conference (RE), IEEE, 2014, pp. 3–12.

[S11] J. L. González Sánchez, R. M. Gil Iranzo, F. L. Gutiérrez Vela, Enriching evaluation in video games, in: IFIP Conference on Human-Computer Interaction, Springer, 2011, pp. 519–522.

[S12] J. L. G. Sánchez, F. L. G. Vela, F. M. Simarro, N. Padilla-Zea, Playability: analysing user experience in video games, Behaviour & Information Technology 31 (10) (2012) 1033–1054.

[S13] S. Vickers, H. Istance, M. Smalley, Eyeguitar: making rhythm based music video games accessible using only eye movements, in: Proceedings of the 7th international conference on advances in computer entertainment technology, 2010, pp. 36–39.

[S14] H. Istance, A. Hyrskykari, L. Immonen, S. Mansikkamaa, S. Vickers, Designing gaze gestures for gaming: an investigation of performance, in: Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications, 2010, pp. 323–330.

[S15] N. Zulfa, D. Yuniasri, P. Damayanti, D. Herumurti, A. A. Yunanto, The effect of ui and ux enhancement on bomberman game based on game experience questionnaire (geq), in: 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), IEEE, 2020, pp. 543–547.

[S16] K. Jørgensen, Between the game system and the fictional world: a study of computer game interfaces, Games and Culture 7 (2) (2012) 142–163.

[S17] J. Nandhakumar, N. S. Panourgias, H. Scarbrough, From knowing it to "getting it": Envisioning practices in computer games development, Information Systems Research 24 (4) (2013) 933–955.

[S18] S. Abrahão, E. Insfran, J. Á. Carsí, A. Fernandez, Early usability in model-driven game development, in: International Conference on Product-Focused Software Process Improvement, Springer, 2016, pp. 713–722.

[S19] Á. Domingo, J. Echeverría, O. Pastor, C. Cetina, Evaluating the benefits of model-driven development - empirical evaluation paper, in: S. Dustdar, E. Yu, C. Salinesi, D. Rieu, V. Pant (Eds.), Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings, Vol. 12127 of Lecture Notes in Computer Science, Springer, 2020, pp. 353–367.

[S20] E. F. do Prado, D. Lucrédio, A flexible model-driven game development approach, in: 2015 IX Brazilian Symposium on Components, Architectures and Reuse Software, IEEE, 2015, pp. 130–139.

[S21] G. W. De Oliveira, S. Julia, L. M. S. Passos, Game modeling using workflow nets, in: 2011 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2011, pp. 838–843.

[S22] F. M. Barreto, S. Julia, Modeling and analysis of video games based on workflow nets and state graphs, in: Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, 2014, pp. 106–119.

[S23] F. M. Barreto, S. JUliA, Formal approach based on petri nets for modeling and verification of video games, Computing and Informatics 40 (1) (2021) 216–248.

[S24] Á. Domingo, J. Echeverría, Ó. Pastor, C. Cetina, Comparing uml-based and dsl-based modeling from subjective and objective perspectives, in: M. La Rosa, S. Sadiq, E. Teniente (Eds.), Advanced Information Systems Engineering, Springer International Publishing, Cham, 2021, pp. 483–498.

[S25] H. Engström, P. A. Östblad, Using text-to-speech to prototype game dialog, Computers in Entertainment (CIE) 16 (4) (2018) 1–16.

[S26] M. J. Best, D. Jacobsen, N. Vining, A. Fedorova, Collection-focused parallelism, in: 5th USENIX Workshop on Hot Topics in Parallelism (HotPar 13), 2013.

[S27] H. Sa'dyah, K. Fathoni, D. K. Basuki, A. Basofi, The fundamental topics of static global illumination algorithms for 3d games, in: 2018 IEEE 3rd international conference on communication and information systems (ICCIS), IEEE, 2018, pp. 237–241.

[S28] H. Xu, C. P. Wang, A review and development of 3-d accelerator technology for games, in: 2009 Second International Symposium on Intelligent Information Technology and Security Informatics, IEEE, 2009, pp. 59–63.

[S29] Y. He, T. Foley, T. Hofstee, H. Long, K. Fatahalian, Shader components: modular and high performance shader development, ACM Transactions on Graphics (TOG) 36 (4) (2017) 1–11.

[S30] V. Stojanovic, D. Blackwood, D. Gilmour, J. P. Isaacs, R. E. Falconer, Comparison of advanced and standard real-time 3d rendering methods for interactive landscapes (short paper version), in: 2013 17th International Conference on Information Visualisation, IEEE, 2013, pp. 539–544.

[S31] K. A. Seitz Jr, T. Foley, S. D. Porumbescu, J. D. Owens, Staged metaprogramming for shader system development, ACM Transactions on Graphics (TOG) 38 (6) (2019) 1–15.

[S32] A. Yacoub, M. E. A. Hamri, C. Frydman, Dev-promela: modeling, verification, and validation of a video game by combining model-checking and simulation, Simulation 96 (11) (2020) 881–910.

[S33] R. Rezin, I. Afanasyev, M. Mazzara, V. Rivera, Model checking in multiplayer games development, in: 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2018, pp. 826–833.

[S34] S. Ding, N. Tang, T. Lin, S. Zhao, Rts-gameflow: a new evaluation framework for rts games, in: 2009 International Conference on Computational Intelligence and Software Engineering, IEEE, 2009, pp. 1–4.

[S35] A. Fernandez, E. Insfran, S. Abrahão, J. Á. Carsí, E. Montero, Integrating usability evaluation into model-driven video game development, in: International Conference on Human-Centred Software Engineering, Springer, 2012, pp. 307–314.

[S36] N. Igawa, T. Yokogawa, M. Takahashi, K. Arimoto, Model checking of visual scripts created by ue4 blueprints, in: 2020 9th International Congress on Advanced Applied Informatics (IIAI-AAI), IEEE, 2020, pp. 512–515.

[S37] S. Morisaki, N. Kasai, K. Kanamori, S. Yamamoto, Detecting source code hotspot in games software using call flow analysis, in: 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2019, pp. 484–489.

[S38] M. Ghoreshi, H. Haghighi, An incremental method for extracting tests from object-oriented specification, Information and Software Technology 78 (2016) 1–26.

[S39] B. R. Sagi, R. Silvestrini, Application of combinatorial tests in video game testing, Quality Engineering 29 (4) (2017) 745–759.

[S40] I. Hasegawa, T. Yokogawa, Automatic verification for node-based visual script notation using model checking, in: International Conference on Formal Engineering Methods, Springer, 2019, pp. 52–68.

[S41] S. Iftikhar, M. Z. Iqbal, M. U. Khan, W. Mahmood, An automated model based testing approach for platform games, in: 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, 2015, pp. 426–435.

[S42] P. Mirza-Babaei, N. Moosajee, B. Drenikow, Playtesting for indie studios, in: Proceedings of the 20th International Academic Mindtrek Conference, 2016, pp. 366–374.

[S43] P. Mirza-Babaei, S. Stahlke, G. Wallner, A. Nova, A postmortem on playtesting: Exploring the impact of playtesting on the critical reception of video games, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–12.

[S44] X. Tong, Positioning game review as a crucial element of game user feedback in the ongoing development of independent video games, Computers in Human Behavior Reports 3 (2021) 100077.

[S45] D. Zagieboylo, K. A. Zaman, Cost-efficient and reliable reporting of highly bursty video game crash data, in: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, 2017, pp. 201–212.

[S46] D. Festa, D. Maggiorini, L. A. Ripamonti, A. Bujari, Supporting distributed real-time debugging in online games, in: 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2017, pp. 737–740.

[S47] A. Truelove, E. S. de Almeida, I. Ahmed, We'll fix it in post: what do bug fixes in video game update notes tell us?, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 736–747.

[S48] C. Lewis, J. Whitehead, Repairing games at runtime or, how we learned to stop worrying and love emergence, IEEE software 28 (5) (2011) 53–59.

[S49] J. Kasurinen, K. Smolander, What do game developers test in their products?, in: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2014, pp. 1–10.

[S50] S. Lynch, K. Pedersen, F. Charles, C. Hargood, M22-a modern visual novel framework, in: Proceedings of the 8th International Workshop on Narrative and Hypertext, 2019, pp. 9–13.

[S51] C. Politowski, F. Petrillo, J. E. Montandon, M. T. Valente, Y.-G. Guéhéneuc, Are game engines software frameworks? a three-perspective study, Journal of Systems and Software 171 (2021) 110846.

[S52] D. MacCormick, L. Zaman, Subvis: the use of subjunctive visual programming environments for exploring alternatives in game development, in: Proceedings of the 14th International Conference on the Foundations of Digital Games, 2019, pp. 1–11.

[S53] T. Nummenmaa, A. Kultima, K. Alha, T. Mikkonen, Applying lehman's laws to game evolution, in: Proceedings of the 2013 International Workshop on Principles of Software Evolution, 2013, pp. 11–17.

[S54] A. Kica, A. La Manna, L. O'Donnell, T. Paolillo, M. Claypool, Nerfs, buffs and bugs-analysis of the impact of patching on league of legends, in: 2016 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2016, pp. 128–135.

[S55] X. Zhong, J. Xu, Game updates enhance players' engagement: a case of dota2, in: 2021 4th International Conference on Information Management and Management Science, 2021, pp. 117–123.

[S56] A. Drachen, A. Canossa, Analyzing spatial user behavior in computer games using geographic information systems, in: Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era, 2009, pp. 182–189.

[S57] T. C. Kohwalter, L. G. P. Murta, E. W. G. Clua, Capturing game telemetry with provenance, in: 2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), IEEE, 2017, pp. 66–75.

[S58] K. Hullett, N. Nagappan, E. Schuh, J. Hopson, Data analytics for game development: Nier track, in: 2011 33rd International Conference on Software Engineering (ICSE), IEEE, 2011, pp. 940–943.

[S59] L. B. Jacob, T. C. Kohwalter, A. F. Machado, E. W. Clua, D. De Oliveira, A non-intrusive approach for 2d platform game design analysis based on provenance data extracted from game streaming, in: 2014 Brazilian Symposium on Computer Games and Digital Entertainment, IEEE, 2014, pp. 41–50.

[S60] Y. Zeleke, J. C. Osborn, R. G. Sanfelice, Analyzing action games: a hybrid systems approach, in: Proceedings of the 14th International Conference on the Foundations of Digital Games, 2019, pp. 1–11.

[S61] K. Hullett, N. Nagappan, E. Schuh, J. Hopson, Empirical analysis of user data in game software development, in: Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, IEEE, 2012, pp. 89–98.

[S62] J. L. Miller, J. Crowcroft, Avatar movement in world of warcraft battlegrounds, in: 2009 8th Annual Workshop on Network and Systems Support for Games (NetGames), IEEE, 2009, pp. 1–6.

[S63] S. Papaloukas, K. Patriarcheas, M. Xenos, Usability assessment heuristics in new genre videogames, in: 2009 13th Panhellenic Conference on Informatics, IEEE, 2009, pp. 202–206.

[S64] P. Sweetser, D. Johnson, P. Wyeth, A. Ozdowska, Gameflow heuristics for designing and evaluating real-time strategy games, in: Proceedings of the 8th Australasian Conference on Interactive Entertainment: Playing the System, 2012, pp. 1–10.

[S65] R. Yanez-Gomez, J. L. Font, D. Cascado-Caballero, J.-L. Sevillano, Heuristic usability evaluation on games: a modular approach, Multimedia Tools and Applications 78 (4) (2019) 4937–4964.

[S66] S. Aleem, L. F. Capretz, F. Ahmed, A digital game maturity model (dgmm), Entertainment Computing 17 (2016) 55–73.

[S67] J. Kasurinen, K. Smolander, Defining an iterative iso/iec 29110 deployment package for game developers, International Journal of Information Technologies and Systems Approach (IJITSA) 10 (1) (2017) 107–125.

[S68] C. M. Kanode, H. M. Haddad, Software engineering challenges in game development, in: 2009 Sixth International Conference on Information Technology: New Generations, IEEE, 2009, pp. 260–265.

[S69] A. Fatima, T. Rasool, U. Qamar, Gdgse: Game development with global software engineering, in: 2018 IEEE Games, Entertainment, Media Conference (GEM), IEEE, 2018, pp. 1–9.

[S70] H. Scarbrough, N. Panourgias, J. Nandhakumar, The role of objects in the coordination of knowledge-intensive projects: A study of computer games development, in: 2012 45th Hawaii International Conference on System Sciences, IEEE, 2012, pp. 4952–4960.

[S71] R. Ramadan, Y. Widyani, Game development life cycle guidelines, in: 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), IEEE, 2013, pp. 95–100.

[S72] N. B. Ahmad, S. A. R. Barakji, T. M. Abou Shahada, Z. A. Anabtawi, How to launch a successful video game: A framework, Entertainment computing 23 (2017) 1–11.

[S73] R. McDaniel, Communication and knowledge management strategies in video game design and development: A case study highlighting key organizational narratives, in: 2015 IEEE International Professional Communication Conference (IPCC), IEEE, 2015, pp. 1–8.

[S74] M. Schmalz, A. Finn, H. Taylor, Risk management in video game development projects, in: 2014 47th Hawaii International Conference on System Sciences, IEEE, 2014, pp. 4325–4334.

[S75] H. Edholm, M. Lidström, J.-P. Steghöfer, H. Burden, Crunch time: The reasons and effects of unpaid overtime in the games industry, in: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), IEEE, 2017, pp. 43–52.

[S76] K. Borowa, A. Zalewski, A. Saczko, Living with technical debt—a perspective from the video game industry, IEEE Software 38 (06) (2021)

65–70.

[S77] J.-W. Liu, C.-Y. Ho, J. Y. Chang, J. C.-A. Tsai, The role of sprint planning and feedback in game development projects: Implications for game quality, Journal of Systems and Software 154 (2019) 79–91.

[S78] M. A. Winget, W. W. Sampson, Game development documentation and institutional collection development policy, in: Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries, 2011, pp. 29–38.

[S79] M. G. Salazar, H. A. Mitre, C. L. Olalde, J. L. G. Sánchez, Proposal of game design document from software engineering requirements perspective, in: 2012 17th International Conference on Computer Games (CGAMES), IEEE, 2012, pp. 81–85.

[S80] E. Guardiola, The gameplay loop: a player activity model for game design and analysis, in: Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology, 2016, pp. 1–7.

[S81] M. S. El-Nasr, Developing games that capture and engage users, in: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, 2019, pp. 9–10.

[S82] W. Scacchi, Modding as an open source approach to extending computer game systems, in: IFIP International Conference on Open Source Systems, Springer, 2011, pp. 62–74.

[S83] M. Suznjevic, I. Stupar, M. Matijasevic, A model and software architecture for mmorpg traffic generation based on player behavior, Multimedia systems 19 (3) (2013) 231–253.

[S84] B. Bryant, H. Saiedian, An evaluation of videogame network architecture performance and security, Computer Networks 192 (2021) 108128.

[S85] M. Kenzel, B. Kerbl, D. Schmalstieg, M. Steinberger, A high-performance software graphics pipeline architecture for the gpu, ACM Transactions on Graphics (TOG) 37 (4) (2018) 1–15.

[S86] A. I. Wang, N. Nordmark, Software architectures and the creative processes in game development, in: International Conference on Entertainment Computing, Springer, 2015, pp. 272–285.

[S87] T. Olsson, D. Toll, A. Wingkvist, M. Ericsson, Evaluation of a static architectural conformance checking method in a line of computer games, in: Proceedings of the 10th international ACM Sigsoft conference on Quality of software architectures, 2014, pp. 113–118.

[S88] T. Ollsson, D. Toll, A. Wingkvist, M. Ericsson, Evolution and evaluation of the model-view-controller architecture in games, in: 2015 IEEE/ACM 4th International Workshop on Games and Software Engineering, IEEE, 2015, pp. 8–14.

[S89] A. Mohebali, T. K. Chiew, Redefining game engine architecture through concurrency, in: International Conference on Intelligent Software Methodologies, Tools, and Techniques, Springer, 2014, pp. 149–161.

[S90] W. K. Mizutani, F. Kon, Unlimited rulebook: A reference architecture for economy mechanics in digital games, in: 2020 IEEE International Conference on Software Architecture (ICSA), IEEE, 2020, pp. 58–68.

[S91] T. Nummenmaa, E. Berki, T. Mikkonen, Exploring games as formal models, in: 2009 Fourth South-East European Workshop on Formal Methods, IEEE, 2009, pp. 60–65.

[S92] A. Srinivasan, V. N. Venkatraman, Architectural convergence and platform evolution: Empirical test of complementor moves in videogames, IEEE Transactions on Engineering Management 67 (2) (2018) 266–282.

[S93] J. Kienzle, A. Denault, Journey: A massively multiplayer online game middleware, IEEE Software 28 (05) (2011) 38–44.

[S94] J. Donkervliet, J. Cuijpers, A. Iosup, Dyconits: Scaling minecraft-like services through dynamically managed inconsistency, in: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), IEEE, 2021, pp. 126–137.

[S95] F. Boaventura, V. T. Sarinho, Mendiga: A minimal engine for digital games, International Journal of Computer Games Technology 2017 (2017).

[S96] B. J. Geisler, F. J. Mitropoulos, S. Kavage, Gamespect: Aspect oriented programming for a video game engine using meta-languages, in: 2019 SoutheastCon, 2019, pp. 1–8.

[S97] B. J. Geisler, S. L. Kavage, A multi-engine aspect-oriented language with modeling integration for video game design, in: R. Ali, H. Kaindl, L. A. Maciaszek (Eds.), Evaluation of Novel Approaches to Software Engineering, Springer International Publishing, Cham, 2021, pp. 336–359.

[S98] F. Nunnari, A. Héloir, Write-Once, Transpile-Everywhere: Re-using Motion Controllers of Virtual Humans Across Multiple Game Engines, 2018, pp. 435–446.

References

[1] T. Wijman, The games market and beyond in 2021: The year in numbers, [Online; accessed 22-December-2021] (2021).
URL `https://newzoo.com/insights/articles/the-games-market-in-2021-the-year-in-numbers-esports-cloud-gaming`

[2] C. Politowski, F. Petrillo, J. E. Montandon, M. T. Valente, Y.-G. Guéhéneuc, Are game engines software frameworks? a three-perspective study, Journal of Systems and Software 171 (2021) 110846.

[3] H. Engström, B. B. Marklund, P. Backlund, M. Toftedahl, Game development from a software and creative product perspective: A quantitative literature review approach, Entertainment Computing 27 (2018) 10–22.

[4] C. Lewis, J. Whitehead, The whats and the whys of games and software engineering, in: Proceedings of the 1st international workshop on games and software engineering, 2011, pp. 1–4.

[5] A. Ampatzoglou, I. Stamelos, Software engineering research for computer games: A systematic review, Information and Software Technology 52 (9) (2010) 888–901.

[6] J. Kasurinen, M. Palacin-Silva, E. Vanhala, What concerns game developers? a study on game development processes, sustainability and metrics, in: 2017 IEEE/ACM 8th Workshop on Emerging Trends in Software Metrics (WETSoM), 2017, pp. 15–21.

[7] J. Kasurinen, Games as software: Similarities and differences between the implementation projects, in: Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, 2016, pp. 33–40.

[8] E. Murphy-Hill, T. Zimmermann, N. Nagappan, Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?, in: Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 1–11.

[9] J. Kasurinen, K. Smolander, What do game developers test in their products?, in: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2014, pp. 1–10.

[10] J. Nandhakumar, N. S. Panourgias, H. Scarbrough, From knowing it to "getting it": Envisioning practices in computer games development, Information Systems Research 24 (4) (2013) 933–955.

[11] T. McKenzie, M. M. Trujillo, S. Hoermann, Software engineering practices and methods in the game development industry, in: Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts, 2019, pp. 181–193.

[12] J. Musil, A. Schweda, D. Winkler, S. Biffl, A survey on the state of the practice in video game software development, Institute of Software Technology and Interactive Systems, Tech. Rep. (2010) 13.

[13] C. Politowski, L. Fontoura, F. Petrillo, Y.-G. Guéhéneuc, Are the old days gone? a survey on actual software engineering processes in video game industry, in: Proceedings of the 5th International Workshop on Games and Software Engineering, 2016, pp. 22–28.

[14] C. Politowski, F. Petrillo, G. C. Ullmann, Y.-G. Guéhéneuc, Game industry problems: An extensive analysis of the gray literature, Information and Software Technology 134 (2021) 106538.

[15] C. Politowski, F. Petrillo, Y.-G. Guéhéneuc, A survey of video game testing, in: 2021 IEEE/ACM International Conference on Automation of Software Test (AST), IEEE, 2021, pp. 90–99.

[16] C. Politowski, F. Petrillo, G. C. Ullmann, J. de Andrade Werly, Y.-G. Guéhéneuc, Dataset of video game development problems, in: Proceedings of the 17th International Conference on Mining Software Repositories, 2020, pp. 553–557.

[17] G. C. Ullmann, C. Politowski, Y.-G. Guéhéneuc, F. Petrillo, J. E. Montandon, Video game project management anti-patterns, in: Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation, 2022, pp. 9–15.

[18] F. Petrillo, M. Pimenta, Is agility out there? agile practices in game development, in: Proceedings of the 28th ACM International Conference on Design of Communication, 2010, pp. 9–15.

[19] M. Washburn Jr, P. Sathiyanarayanan, M. Nagappan, T. Zimmermann, C. Bird, What went right and what went wrong: an analysis of 155 postmortems from game development, in: Proceedings of the 38th International Conference on Software Engineering Companion, 2016, pp. 280–289.

[20] W. K. Mizutani, V. K. Daros, F. Kon, Software architecture for digital

game mechanics: A systematic literature review, Entertainment Computing 38 (2021) 100421.
[21] J. R. M. Viana, N. P. Viana, F. A. M. Trinta, W. V. De Carvalho, A systematic review on software engineering in pervasive games development, in: 2014 Brazilian Symposium on Computer Games and Digital Entertainment, IEEE, 2014, pp. 51–60.
[22] M. Zhu, A. I. Wang, Model-driven game development: A literature review, ACM Computing Surveys (CSUR) 52 (6) (2019) 1–32.
[23] C. Politowski, Y.-G. Guéhéneuc, F. Petrillo, Towards automated video game testing: still a long way to go, in: Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation, 2022, pp. 37–43.
[24] A. Osborne O'Hagan, G. Coleman, R. V. O'Connor, Software development processes for games: A systematic literature review, in: European Conference on Software Process Improvement, Springer, 2014, pp. 182–193.
[25] J. A. Vargas, L. García-Mundo, M. Genero, M. Piattini, A systematic mapping study on serious game quality, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, 2014, pp. 1–10.
[26] S. Aleem, L. F. Capretz, F. Ahmed, Game development software engineering process life cycle: a systematic review, Journal of Software Engineering Research and Development 4 (1) (2016) 1–30.
[27] S. Aleem, L. F. Capretz, F. Ahmed, A digital game maturity model (dgmm), Entertainment Computing 17 (2016) 55–73.
[28] K. Borowa, A. Zalewski, A. Saczko, Living with technical debt—a perspective from the video game industry, IEEE Software 38 (06) (2021) 65–70.
[29] J. Kasurinen, A. Maglyas, K. Smolander, Is requirements engineering useless in game development?, in: Requirements Engineering: Foundation for Software Quality: 20th International Working Conference, REFSQ 2014, Essen, Germany, April 7-10, 2014. Proceedings 20, Springer, 2014, pp. 1–16.
[30] F. Ahmed, M. Zia, H. Mahmood, S. Al Kobaisi, Open source computer game application: An empirical analysis of quality concerns, Entertainment Computing 21 (2017) 1–10.
[31] C. M. Kanode, H. M. Haddad, Software engineering challenges in game development, in: 2009 Sixth International Conference on Information Technology: New Generations, IEEE, 2009, pp. 260–265.
[32] D. Callele, E. Neufeld, K. Schneider, Requirements engineering and the creative process in the video game industry, in: 13th IEEE International Conference on Requirements Engineering (RE'05), IEEE, 2005, pp. 240–250.
[33] L. Pascarella, F. Palomba, M. Di Penta, A. Bacchelli, How is video game development different from software development in open source?, in: 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), IEEE, 2018, pp. 392–402.
[34] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering (2007).
[35] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, Journal of systems and software 80 (4) (2007) 571–583.
[36] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Information and software technology 51 (1) (2009) 7–15.
[37] V. Alves, N. Niu, C. Alves, G. Valença, Requirements engineering for software product lines: A systematic literature review, Information and Software Technology 52 (8) (2010) 806–820.
[38] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele University 33 (2004) (2004) 1–26.
[39] A. for Computing Machinery, The 2012 acm computing classification system, [Online; accessed 07-June-2023] (2012).
URL https://www.acm.org/publications/class-2012
[40] R. L. Glass, I. Vessey, V. Ramesh, Research in software engineering: an analysis of the literature, Information and Software technology 44 (8) (2002) 491–506.
[41] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, 2000.
[42] M. Felderer, G. H. Travassos, Contemporary Empirical Methods in Software Engineering, Springer, 2020.
[43] K. Mao, L. Capra, M. Harman, Y. Jia, A survey of the use of crowdsourcing in software engineering, Journal of Systems and Software 126 (2017) 57–84.
[44] T. Dyba, T. Dingsoyr, G. K. Hanssen, Applying systematic reviews to diverse study types: An experience report, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), IEEE, 2007, pp. 225–234.
[45] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Information and software technology 55 (12) (2013) 2049–2075.
[46] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and software technology 64 (2015) 1–18.
[47] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th international conference on evaluation and assessment in software engineering, 2014, pp. 1–10.
[48] M.-T. Cheng, J.-H. Chen, S.-J. Chu, S.-Y. Chen, The use of serious games in science education: a review of selected empirical research from 2002 to 2013, Journal of computers in education 2 (3) (2015) 353–375.
[49] G. N. Yannakakis, J. Togelius, Artificial intelligence and games, Vol. 2, Springer, 2018.
[50] B. A. Kitchenham, O. P. Brereton, D. Budgen, Z. Li, An evaluation of quality checklist proposals-a participant-observer case study, in: 13th International Conference on Evaluation and Assessment in Software Engineering (EASE) 13, 2009, pp. 1–10.
[51] M. Galster, D. Weyns, D. Tofan, B. Michalik, P. Avgeriou, Variability in software systems—a systematic literature review, IEEE Transactions on Software Engineering 40 (3) (2013) 282–306.
[52] M. S. Ali, M. A. Babar, L. Chen, K.-J. Stol, A systematic review of comparative evidence of aspect-oriented programming, Information and software Technology 52 (9) (2010) 871–887.
[53] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and software technology 50 (9-10) (2008) 833–859.
[54] A. Fung, The impact of the rise of mobile games on the creativity and structure of the games industry in china, Mobile Gaming in Asia: Politics, Culture and Emerging Technologies (2017) 91–103.
[55] Ó. P. Latorre, et al., Indie or mainstream? tensions and nuances between the alternative and the mainstream in indie games, Anàlisi (2016) 15–30.
[56] M. McShaffry, Game coding complete, Cengage Learning, 2009.
[57] F. Vaudour, A. Heinze, Software as a service: Lessons from the video game industry, Global Business and Organizational Excellence 39 (2) (2020) 31–40.
[58] M. S. O. Almeida, F. S. C. da Silva, A systematic review of game design methods and tools, in: Entertainment Computing–ICEC 2013: 12th International Conference, ICEC 2013, São Paulo, Brazil, October 16-18, 2013. Proceedings 12, Springer, 2013, pp. 17–29.
[59] C. Calero, M. Polo, M. Á. Moraga, Investigating the impact on execution time and energy consumption of developing with spring, Sustainable Computing: Informatics and Systems 32 (2021) 100603.
[60] T. E. Colanzi, W. K. Assunção, S. R. Vergilio, P. R. Farah, G. Guizzo, The symposium on search-based software engineering: Past, present and future, Information and Software Technology 127 (2020) 106372.
[61] F. Pérez, J. Font, L. Arcega, C. Cetina, Empowering the human as the fitness function in search-based model-driven engineering, IEEE Transactions on Software Engineering (01) (2021) 1–1.
[62] C. Wohlin, M. Höst, K. Henningsson, Empirical research methods in software engineering, in: Empirical methods and studies in software engineering, Springer, 2003, pp. 7–23.
[63] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer Science & Business Media, 2012.
[64] B. A. Kitchenham, D. Budgen, P. Brereton, Evidence-based software engineering and systematic reviews, Vol. 4, CRC press, 2015.
[65] S. Shuermans, C. Voskoglou, Global developer population report 2019., https://www.slashdata.co/?link=%20GlobalDevPop19, [Online; accessed 13-December-2022] (2019).

APPENDIX

Table 6: Previous Study Data-set after meeting our selection criteria.

#	Publisher	Citation Type	CCS'98	Country	Year	Method	Approach
S04	IEEE	Conference	D.2.9	Netherlands	2009	-	Exploratory
S06	Elsevier	Journal	D.2.5	Netherlands	2007	Experiment	Empirical
S07	Springer	Conference	D.2.0	Finland	2002	-	Descriptive
S09	ACM	Journal	D.2.0	-	2004	-	Descriptive
S10	ACM	Conference	D.2.1	UK	2009	-	Exploratory
S11	IEEE	Conference	D.2.1	Canada	2005	Case Study	Empirical
S18	ACM	Conference	D.2.1	USA	2004	-	Exploratory
S19	Elsevier	Journal	D.2.3	Canada	2007	-	Exploratory
S29	Springer	Workshop	D.2.5	Canada	2007	-	Descriptive
S31	Elsevier	Journal	D.2.1	UK	2005	Survey	Empirical
S34	IEEE	Conference	D.2.9	USA	2009	-	Descriptive
S40	Springer	Conference	D.2.6	Korea	2004	-	Descriptive
S44	IEEE	Conference	D.2.6	Canada	2004	-	Exploratory
S50	ACM	Conference	D.2.1	Finland	2009	-	Descriptive
S53	ACM	Conference	D.2.9	Finland	2008	-	Descriptive
S54	IEEE	Conference	D.2.1	China	2009	Survey	Empirical
S55	ACM	Journal	D.2.9	Brazil	2009	Survey	Empirical
S57	ACM	Conference	D.2.5	USA	2008	-	Exploratory
S58	ACM	Conference	D.2.1	USA	2008	Survey	Empirical
S70	IEEE	Conference	D.2.9	UK	2005	-	Exploratory
S71	ACM	Journal	D.2.3	USA	2008	-	Exploratory
S73	Springer	Conference	D.2.1	Netherlands	2009	Experiment	Empirical
S76	ACM	Journal	D.2.1	USA	2008	-	Exploratory
S79	ACM	Journal	D.2.3	USA	2009	-	Exploratory

Table 7: Study Data-set

#	Publisher	Citation Type	CCS'98	CCS'12	Country	Year	Method	Approach	QA
S1	IEEE	Workshop	D.2.0	5.3.4	USA	2016	Case study	Empirical	3
S2	SciTePress	Conference	D.2.0	5.3.4	USA	2021	Case study	Empirical	2.5
S3	ACM	Workshop	D.2.1	5.2.2	Netherlands	2012	-	Descriptive	1.5
S4	Springer	Journal	D.2.1	5.2.5	Spain	2020	Case study	Empirical	3
S5	Springer	Conference	D.2.1	5.1.1	Spain	2020	Case study	Empirical	3
S6	Springer	Conference	D.2.1	5.2.5	Spain	2021	Case study	Empirical	2.5
S7	Springer	Conference	D.2.1	5.3.1	Finland	2014	Survey	Empirical	4.5
S8	ACM	Conference	D.2.1	5.3.1	Finland	2013	Survey	Empirical	5.5
S9	ACM	Journal	D.2.1	5.3.1	UK	2021	Survey	Empirical	5
S10	IEEE	Conference	D.2.1	5.3.1	Netherlands	2014	Survey	Empirical	5
S11	Springer	Conference	D.2.2	5.3.1	Spain	2011	Experiment	Empirical	3
S12	Taylor & Francis Group	Journal	D.2.2	5.3.1	Spain	2012	-	Descriptive	3
S13	ACM	Conference	D.2.2	5.1.1	UK	2010	Case study	Empirical	4
S14	ACM	Conference	D.2.2	5.3.1	UK	2010	Experiment	Empirical	4
S15	IEEE	Conference	D.2.2	5.1.1	Indonesia	2020	Survey	Empirical	4
S16	SAGE	Journal	D.2.2	5.1.1	Norway	2012	Experiment	Empirical	1
S17	INFORMS	Journal	D.2.2	5.3.3	UK	2013	Survey	Empirical	4
S18	Springer	Conference	D.2.2	5.1.4	Spain	2016	-	Descriptive	3.5
S19	Springer	Conference	D.2.2	5.1.2	Spain	2020	Experiment	Empirical	6

Table 7: Study Data-set (continued)

#	Publisher	Citation Type	CCS'98	CCS'12	Country	Year	Method	Approach	QA
S20	IEEE	Conference	D.2.2	5.3.3	Brazil	2015	Experiment	Empirical	6
S21	IEEE	Conference	D.2.2	5.1.2	Brazil	2011	-	Descriptive	1.5
S22	IBM	Conference	D.2.2	5.1.2	Brazil	2014	-	Descriptive	3.5
S23	Slovak Academy of Sciences	Journal	D.2.2	5.1.2	Brazil	2021	-	Descriptive	2.5
S24	Springer	Conference	D.2.2	5.2.4	Spain	2021	Experiment	Empirical	6
S25	ACM	Journal	D.2.2	5.3.1	Sweden	2018	Case study	Empirical	4.5
S26	USENIX	Workshop	D.2.3	5.3.1	Canada	2013	Case study	Empirical	1.5
S27	IEEE	Conference	D.2.3	5.3.3	Indonesia	2019	-	Descriptive	3
S28	IEEE	Conference	D.2.3	5.3.3	China	2009	-	Descriptive	2.5
S29	ACM	Journal	D.2.3	5.3.3	USA	2017	Case study	Empirical	5
S30	IEEE	Conference	D.2.3	5.3.3	Germany	2013	Survey	Empirical	2
S31	ACM	Journal	D.2.3	5.2.4	USA	2019	Case study	Empirical	4.5
S32	SAGE	Journal	D.2.4	5.3.4	France	2020	Case study	Empirical	4
S33	IEEE	Conference	D.2.4	5.3.4	Russia	2018	-	Descriptive	3.5
S34	IEEE	Conference	D.2.4	5.3.4	China	2009	Experiment	Empirical	1.5
S35	Springer	Conference	D.2.4	5.1.3	Spain	2012	-	Descriptive	2.5
S36	IEEE	Conference	D.2.4	5.1.3	Japan	2020	-	Descriptive	0.5
S37	IEEE	Conference	D.2.4	5.3.4	Japan	2019	Case study	Empirical	3
S38	Elsevier	Journal	D.2.5	5.3.4	Iran	2016	-	Descriptive	3.5
S39	Taylor & Francis Group	Journal	D.2.5	5.3.4	USA	2017	Case study	Empirical	4
S40	Springer	Conference	D.2.5	5.3.4	Japan	2019	-	Descriptive	3.5
S41	IEEE	Conference	D.2.5	5.3.4	Pakistan	2015	Case study	Empirical	4.5
S42	ACM	Conference	D.2.5	5.3.4	Canada	2016	Case study	Empirical	3.5
S43	ACM	Conference	D.2.5	5.3.4	Canada	2020	Case study	Empirical	3
S44	Elsevier	Journal	D.2.5	5.3.4	UK	2021	Case study	Empirical	5
S45	ACM	Conference	D.2.5	5.3.4	USA	2017	-	Descriptive	4
S46	IEEE	Conference	D.2.5	5.3.4	Italy	2017	Case study	Empirical	2
S47	IEEE	Conference	D.2.5	5.3.4	USA	2021	-	Descriptive	6
S48	IEEE	Journal	D.2.5	5.3.4	USA	2011	Case study	Empirical	3
S49	ACM	Conference	D.2.5	5.3.2	Finland	2014	Survey	Empirical	6
S50	ACM	Workshop	D.2.6	5.2.6	UK	2019	Case study	Empirical	3
S51	Elsevier	Journal	D.2.6	5.2.6	Canada	2021	Survey	Empirical	4.5
S52	ACM	Conference	D.2.6	5.2.4	Canada	2019	Experiment	Empirical	3.5
S53	ACM	Workshop	D.2.7	5.3.5	Finland	2013	-	Descriptive	2
S54	IEEE	Conference	D.2.7	5.3.5	USA	2016	Case study	Empirical	3
S55	ACM	Journal	D.2.7	5.3.5	China	2021	Experiment	Empirical	4
S56	ACM	Conference	D.2.8	5.3.4	Denmark	2009	Case study	Empirical	4
S57	IEEE	Conference	D.2.8	5.3.4	Brazil	2018	Case study	Empirical	1.5
S58	IEEE	Conference	D.2.8	5.3.4	USA	2011	Case study	Empirical	4
S59	IEEE	Conference	D.2.8	5.3.4	Brazil	2014	-	Descriptive	3
S60	ACM	Conference	D.2.8	5.3.4	USA	2019	Case study	Empirical	2.5
S61	IEEE	Conference	D.2.8	5.3.4	USA	2012	Case study	Empirical	4.5
S62	IEEE	Workshop	D.2.8	5.3.4	UK	2009	-	Descriptive	2
S63	IEEE	Conference	D.2.8	5.1.4	Greece	2009	Case study	Empirical	3.5
S64	ACM	Conference	D.2.8	5.3.4	Australia	2012	Survey	Empirical	2.5
S65	Springer	Journal	D.2.8	5.3.4	Spain	2019	-	Descriptive	4
S66	Elsevier	Journal	D.2.9	5.3.2	Canada	2016	-	Descriptive	6
S67	IGI Global Publishing	Journal	D.2.9	5.3.2	Finland	2017	Survey	Empirical	4.5
S68	IEEE	Conference	D.2.9	5.3.2	USA	2009	-	Exploratory	1
S69	IEEE	Conference	D.2.9	5.3.2	Pakistan	2018	-	Descriptive	1

Table 7: Study Data-set (continued)

#	Publisher	Citation Type	CCS'98	CCS'12	Country	Year	Method	Approach	QA
S70	IEEE	Conference	D.2.9	5.3.2	UK	2012	Survey	Empirical	4.5
S71	IEEE	Conference	D.2.9	5.3.2	Indonesia	2013	Case study	Empirical	2.5
S72	Elsevier	Journal	D.2.9	5.3.2	United Arab Emirates	2017	-	Descriptive	3.5
S73	IEEE	Conference	D.2.9	5.3.2	USA	2015	Survey	Empirical	4
S74	IEEE	Conference	D.2.9	5.3.2	USA	2014	Survey	Empirical	4
S75	IEEE	Conference	D.2.9	5.3.2	Sweden	2017	Survey	Empirical	6
S76	IEEE	Journal	D.2.9	5.3.2	Poland	2021	Survey	Empirical	4
S77	Elsevier	Journal	D.2.9	5.3.2	Taiwan	2019	Survey	Empirical	5.5
S78	ACM	Conference	D.2.10	5.3.1	USA	2011	-	Descriptive	4
S79	IEEE	Conference	D.2.10	5.3.1	Mexico	2012	Case study	Empirical	5
S80	ACM	Conference	D.2.10	5.3.1	Germany	2016	Case study	Empirical	2.5
S81	IEEE	Conference	D.2.10	5.3.1	USA	2019	Case study	Empirical	0.5
S82	IGI Global Publishing	Journal	D.2.10	5.3.1	USA	2011	-	Descriptive	1.5
S83	Springer	Journal	D.2.11	5.1.2	Croatia	2013	Case study	Empirical	4
S84	Elsevier	Journal	D.2.11	5.1.2	USA	2021	-	Descriptive	3
S85	ACM	Journal	D.2.11	5.1.2	Austria	2018	Experiment	Empirical	3
S86	Springer	Journal	D.2.11	5.1.2	Norway	2015	Survey	Empirical	4
S87	ACM	Conference	D.2.11	5.1.2	Sweden	2014	-	Descriptive	4
S88	IEEE	Workshop	D.2.11	5.1.2	Sweden	2015	Case study	Empirical	6
S89	Springer	Journal	D.2.11	5.1.2	Malaysia	2014	Survey	Empirical	3.5
S90	IEEE	Conference	D.2.11	5.1.2	Brazil	2020	Experiment	Empirical	5
S91	IEEE	Workshop	D.2.11	5.3.5	Finland	2009	-	Descriptive	2
S92	IEEE	Journal	D.2.11	5.1.2	USA	2020	-	Descriptive	2.5
S93	IEEE	Journal	D.2.12	5.1.4	Canada	2011	Case study	Empirical	4
S94	IEEE	Journal	D.2.12	5.1.2	Netherlands	2021	Case study	Empirical	5
S95	Hindawi	Journal	D.2.13	5.2.6	Brazil	2017	-	Descriptive	2.5
S96	IEEE	Conference	D.2.13	5.3.3	USA	2019	Case study	Empirical	3
S97	Springer	Conference	D.2.13	5.3.3	USA	2021	Case study	Empirical	3
S98	Springer	Conference	D.2.13	5.3.3	Germany	2018	Case study	Empirical	3