Computer Science Review 47 (2023) 100517

Contents lists available at ScienceDirect

Computer Science Review

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

Review article

Software as storytelling: A systematic literature review

Paolo Ciancarini [a,b], Mirko Farina [a], Ozioma Okonicha [a], Marina Smirnova [a], Giancarlo Succi [b,*]

[a] Innopolis University, Russia
[b] University of Bologna, Italy

ARTICLE INFO

Article history:
Received 15 June 2021
Received in revised form 21 October 2022
Accepted 5 November 2022
Available online 26 November 2022

Keywords:
Storytelling
Software development
Story-driven approach
Coding
Story-based techniques

ABSTRACT

Storytelling has always been a crucial, perhaps constitutive part of our lives. All communities have told stories. In recent years, software development is becoming increasingly recognized as a creative process that has a lot in common with the process of writing or telling a story. *Aim*: The objectives of this paper are: **(a)** to review and aptly classify current principles and approaches that describe software development as a form of storytelling; **(b)** to describe and understand the heuristic function of storytelling in software development; and **(c)** to discuss and single out the principles of storytelling that may play a role, hence constitute significant improvements to the practices of software developers.

Method: To achieve these goals and objectives we conducted a systematic literature review of relevant scientific papers and subsequently analyzed them by means of a textual narrative synthesis.

Results: More specifically, we retrieved, screened and examined 51 relevant publications. The synthesis we conducted allowed us to understand and better visualize the many interesting correspondences and analogies between those two seemingly different processes, namely storytelling and software development. In particular, in our work, we focused on describing and analyzing how certain principles underlying storytelling can be adapted and applied in current practices of software engineering.

Conclusion: This paper presents and re-elaborates in a critical fashion and from a different angle a substantial body of knowledge and research recently carried out in the software development literature.

Contents

1. Introduction ... 2
   1.1. Models of storytelling ... 2
   1.2. Software development and storytelling ... 3
      1.2.1. Basic practices of storytelling ... 3
      1.2.2. Basic practices of software development ... 4
      1.2.3. Similarities between these approaches and benefits for software development ... 4
   1.3. Motivations, aims, and goals of the present work ... 5
2. Research methodology ... 5
   2.1. PRISMA checklist ... 6
   2.2. Research questions ... 6
   2.3. Search strategy ... 6
   2.4. Search sources and query construction ... 6
   2.5. Inclusion and exclusion criteria ... 7
   2.6. Search process ... 7
   2.7. Selection process ... 7
   2.8. Data collection process ... 8
   2.9. Data analysis ... 8
3. Results ... 9

* Corresponding author.
E-mail address: g.succi@unibo.it (G. Succi).

https://doi.org/10.1016/j.cosrev.2022.100517
1574-0137/

3.1. Data collected ... 9
3.2. RQ1: What are the similarities between the practice of software development and the practice of storytelling? ... 9
3.3. RQ2: What are the specific goals of storytelling, as perceived by software engineers, during software development? ... 10
3.4. RQ3: What are the principles underlying storytelling that can help in refining current practices in software development? ... 14
4. Discussion ... 15
4.1. RQ1: What are the similarities between the practice of software development and the practice of story-telling? ... 15
4.2. RQ2: What are the purposes of storytelling, especially as perceived by software engineers, during software development? ... 16
4.3. RQ3: What are the principles underlying storytelling that can help in refining current practices in software development? ... 16
5. Limitations, threats to validity and review assessment ... 16
5.1. Limitations ... 16
5.2. Threats to validity ... 17
5.3. Review assessment ... 17
6. Conclusion ... 17
Declaration of competing interest ... 18
Data availability ... 18
Acknowledgments ... 18
Appendix ... 18
References ... 18

1. Introduction

Since antiquity, storytelling has played an essential role in our lives as a tool for education [1], communication [2], knowledge management [3], and cultural transmission [4], as well as for entertainment purposes [5].

It can be argued that storytelling is an essential part of our lives, an intrinsic feature of us humans. We are engaged in this activity nearly all the time — while studying, working, watching TV, chatting with people. However, we pay little attention on how such an activity influences us and the world around us. Even its definition is not fully formalizable because it is a strongly cross-disciplinary concept [6,7].

Roughly speaking, storytelling is the activity of passing to other people ideas, beliefs, or experiences of various kinds. This activity is typically performed through telling or writing stories. A story can be described as a run-down of actions gathered from events, either factual or fantasized [8].

The way in which such an activity or practice has been used to communicate with others has dramatically evolved over the course of time [9]. One may argue that storytelling originated with visual stories, such as cave drawings dating back 30 000 years ago (think about the Chauvet cave or the Lascaux cave in France) [10], and then shifted to oral traditions, in which stories were passed cross and trans-generationally by word of mouth (think about Aboriginal practices of storytelling involving – for instance – songs, which were sang as mnemonic devices to successfully navigate in barren desert environments) [11]. From that time, one may notice, a slow but steady shifts towards more coherent and systematized narratives (including written, printed, typed and even [more recently] digital stories).

1.1. Models of storytelling

There are many different ways to tell a story — hence, different models of storytelling, each of which depends on a variety of factors including the goals that one wants to accomplish, the audience that one wants to reach, and/or the time who has to convey a certain story's message [12]. For example, in the business industry the most reputable models of storytelling are Miller's StoryBrand [13], McKee's Storynomics [14], and Raskin's Strategic Narrative Model[1]

In our review of the relevant literature we found that the following are the most widely used types (or forms) of storytelling:

A. Oral Storytelling [15];
B. Written Storytelling [16];
C. Storytelling through Artistic Performances (typically via dance and/or music) [17,18];
D. Visual Storytelling (through – for instance – paintings and/or drawings) [19];
E. Digital Storytelling [20];
F. Virtual Reality Storytelling [21];
G. Transmedia Storytelling [22]

There are also several techniques that can be adopted to tell a story. These include: i. a linear narrative, ii. a non-linear narrative, iii. a quest narrative, iv. a viewpoint narrative, which can all be used with different goals and purposes (such as entertaining [23], instructing [24], persuading, Jones and Peterson [25], moralizing [26], etc.).

Since in this paper we are interested in the role of storytelling in software development, we focus mainly on textual storytelling, which can either be oral or written, or both and which is usually carried out via linear or non-linear narratives. According to [27] it is possible to distinguish between two different models of text composition in storytelling: a stage process model and a cognitive process model.

The former sees composition as a sequence of precise and consequential stages, which unfold in a specific order in accordance to a linear narrative. Examples of this type of model include: the "Pre-Write/Write/Re-Write" model by [28] and the Huizenga model, more on which below [29].

The cognitive process model, unlike the stage process model, does not require the linear and sequential completion of predetermined steps or stages; rather, it envisages the possibility of running processes without a specific hierarchical order, in a non-linear way. The idea is therefore that the processes of the model ought not be treated as guidelines to be strictly followed in order to compose and write well; rather, as flexible, fluid indications for describing how people may write. [27] describe three basic processes underlying such a model. These are: "Planning, Translating, and Reviewing". Through Planning, one typically generates ideas, sets goals, and processes information. By Translating one expresses ideas and goals in some kind of language. Finally, through Reviewing one evaluates and critically assesses the work performed.

Thus, on the one hand, we can view storytelling as a creative activity emerging from non-linear narratives, often requiring a significant amount of resources, very powerful imagination, and a bit of an "instinct for narrative" [30]. On the other hand, though, we can also consider storytelling as a skill that can be effectively

[1] https://andyraskin.com/.

developed and mastered only through years of rigorous practice and strict discipline [31, pp. 15], [32,33].

While there can be very many components in a story e.g., the core values, the hero, the mentor, the context, the conflict, the change, the point of view, the resolution; [13] a story's set of minimal or essential elements is usually restricted to four main ones: the plot, the characters, the message and the conflict [34]. Naturally, depending on the context and practices involved more extended forms of storytelling can be produced or envisaged, involving any combination of the elements aforementioned. For example, in journalism the *5 W approach* addresses the fundamental questions ['Who', 'What', 'Where', 'When', and 'Why'] that every story should be able to answer.

The fact that stories can possess many different elements, raises the question of which (if any) of them is more important. [34] argued that the message is the most important element in any story. Other people disagree. For example, House of Cards showrunner Beau Willimon famously claimed that the conflict is the most important element in any story. [35] argued instead that change is the most crucial component of a story as "many stories begin with a moment of unexpected change". Change is both a promise of opportunities and a dangerous path. According to Storr's, storytellers purposefully create scenarios for unexpected changes through which they seize the attention of both their protagonists and their readers or listeners.

Regardless of which element is the most crucial one in a story, we can say that storytelling at minimum necessarily involves at least two actors: a teller/a writer and a listener, but possibly several tellers and several listeners. So, in general we can say that storytelling demands a community of people creating and sharing a story. An example is the novel [36], which was written to support a new approach to collaboration among software developers. This example supports drawing a preliminary yet direct link between storytelling and an apparently completely different activity, namely that of software development as conducted by a team of programmers [37].

1.2. Software development and storytelling

Consider next the definition of software development given in the Standard Glossary of Software Engineering Terminology (1990) [38], which roughly corresponds to what Kernighan and Plauger (1976) wrote a decade before [39] that states that: software development is "the process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use". The definition is important because it characterizes the software as a mutable entity comprising different kinds of artifacts, such as: "computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system" [38].

As for storytelling there are many different models of software development.[2] Typically researchers distinguish models of software development in accordance to how they approach the workflow organization (whether linearly or iteratively) and on the basis of the relationships that are established between the development team and the end customer. Broadly speaking, we can say that there are two major categories of models. Those that require low customer involvement (such as Waterfall-type models) and those that require more cooperation, hence are collaborative in character (such as the Rational Unified Process). Next, we briefly discuss these two examples as generic illustration of this point.

Waterfall-type models, which fall into the more general category of system development life cycle (SDLC) models [40], tend to describe the development project in terms of fix sequences of – strictly documented – actions (such as analysis, design, coding, testing, and implementation) that determine concrete deliverables [41]. Such models are typically used when there is a need for strict control over the project; for example when there is a limited budged and/or hard deadlines.

The Rational Unified Process [42] is an agile development method that combines both linear and iterative components. The model is characterized by a series of basic activities (such as requirement, design, implementation, testing etc.) that are developed in parallel across different phases through different level of intensity. This model is typically used in large and/or high-risk projects where there is a need of fast development of high-quality items [42].

Here we would like to note the *transformational* and *relational* nature of software development that corresponds to the *transformational* and *relational* nature of storytelling. We also note that the many different modes of telling a story are reflected in the wide variety of ways in which software is developed [7,43].

> The questions that naturally arise are then:
> i. how does software development relate to the practice of storytelling?, and ii. what we can learn from this relationship [43]?.
> This paper explores how current research has addressed such questions.

Prima facie, one can argue that one needs to be creative in order to find innovative and effective solutions in software development [44,45]. This sort of creativity, however, is also dependent upon a number of conditions, including technical, economic, and even environmental or socio-cultural factors [46–49]. One, however, can also say that creativity in software development often involves finding the appropriate paths for handling customers' requirements, or selecting/adopting/combining or even making up new solutions and strategies to existing problems [50,51].

In an attempt to further specify the parallel between storytelling and software development that we introduced above, we next focus on analyzing some of the key elements of these practices, trying to map a one-to-one correspondence that could shed light on how the identification of basic elements in a story could play a beneficial role in software development. We note that the elements we describe and summarize below emerged from the iterative analysis and interpretation of the data we reviewed (more on which below).[3]

1.2.1. Basic practices of storytelling

A. "Choosing the Script" - "The script or plot is the structure that holds the story together and defines the sequence of actions that unfold. This is a crucial element for formulation and exposition of any story";

B. "Determining the Level of Detail" – "The level of detail of a story represents a crucially important factor in the practice of storytelling. Including many details (hence providing comprehensive accounts of events) may be surely tempting; however, it may come at cost; that of confusing or boring the target audience. For this reason, in order to achieve the right level of detail in any given story it is customary to appeal to "Elevator Pitches"; practices in which

[2] Again, it is beyond the scope of this paper to provide a full account of all those different approaches, so what follows is only a brief primer for the reader. For more details please refer to: https://www.scnsoft.com/blog/software-development-models.

[3] Thanks to the reviewer for pressing us on this point.

storytellers include the minimal details that are necessary to understand the key idea in as little time as an elevator ride".;

C. "Paired storytelling" - "Two students are paired to read together a story, that then is summarized (written or told) cooperatively" [52].
D. "Engaging with the Audience" – "it is an important feature of storytelling because engagement often determines the success or failure (hence, the transmissibility) of a particular story" [8].

1.2.2. Basic practices of software development

Software development is a problem solving activity performed through the application of principles, concepts, and tools. Software practices are made to ease the process of software development by providing patterns on how to act or think in given conditions. For instance, Kent Beck's Extreme Programming (in short, XP) is a well-known agile development method which includes several specific practices [53]. Below, we present examples of some of the practices recommended by Beck and included in XP:

a. "The Planning Game" - this practice aims at distributing responsibilities between business and developers. The former typically decides on the importance and priority of a feature, whereas the latter settles the costs. This practice is designed to maximize efficiency of development as well as the productive interactions among the parties involved.
b. "Acceptance Tests" - A practice which recommends to ask the business to test and validate each release of the product. User stories cannot capture in advance all the details required by the stakeholders. These tests are needed to verify whether the product is ultimately compliant with the stakeholders' requirements.
c. "Pair Programming" - A practice whereby two developers use only one computer and one keyboard to write a program cooperatively.
d. "Customer Team Member" - business and development sides work together continuously on a daily basis. This means that the customer is readily accessible to the developers who can ask for immediate feedback when they have a doubt [54].

1.2.3. Similarities between these approaches and benefits for software development

We next discuss some of the correspondences we found across the basic elements underlying the practices we described above.

i. "The Planning Game" aka "Choosing the Script" – both processes need a practice that allows for careful planning of future activities, before the implementation is carried out [8];
ii. "Acceptance Tests" aka "Determining the Level of Detail" – for both activities it is absolutely crucial to decide, which details must be included so that steady progress can be achieved;
iii. "Paired Storytelling" aka "Pair programming" – people cooperate to improve their product, by checking independently its quality;
iv. "Customer Team Member" aka "Engaging with the Audience" – if we consider developers as storytellers, then the audience to whom they must relate is the customer. So, developers need to learn the etiquette (habits) for communicating with business people as well as a set of techniques that may favor the adequate presentation (and implementation) of their work.

We could also look at the potential similarities between these two activities – storytelling and software development – from a more general point of view. If we do so, we quickly realize that writing or telling stories and designing software programs have much in common. We remark that in both practices certain characteristics of the end products are similar. For instance, Komyakhov et al. (2020) used writing stories as a proxy for coding teaching principles of lean software development [55]. Moreover, Hermans and Aldewereld (2017) performed a precise mapping of the steps involved in writing or telling a story and compared that with the steps typically required in programming or coding [56]. They discovered that both processes require gathering of information, planning the work to be done, translating abstract ideas into concrete representations and reviewing and honing the final results etc. In particular, during the initial stages of software design, various practices and principles underlying storytelling can be actively and effectively used to:

1. refine planning analyzing the stories and refactoring their structure [57],
2. facilitate requirements elicitation and decide which details are important [58–60], and
3. improve prototyping to engage the customers and users [8].

With respect to common structural elements between these two processes and sets of practices, Jones et al. (2011) described similarities between the English grammar and structure and a computer program [61]. Further, metaphors have been widely taken from the literary domain and effectively adopted to describe concepts and processes of the IT and programming domains [62,63]. A structural analysis and comparison is also possible using semiotics: for instance, Herrenschmidt (2007) compared the cognitive structures recruited for writing with those recruited by mathematics and coding [64].

Having spelled out (at least some of) the correspondences between these two practices we are now in a position to briefly reflect on how the elements of the former (storytelling) may have a positive impact on the latter (software development). Software developers actually use some form of storytelling in a variety of situations: for instance, during the requirements phase they can collect stories and scenarios to help clarifying the needs of the stakeholders. Similarly, during the design phase developers often use architectural metaphors and design patterns in form of stories or shared vocabularies. In addition, during the construction phase they can use visual stories to help visualizing the evolution of the codebase. Furthermore, during the verification phase developers may use scenarios and stories to support code inspections, and so on.

In truth, from the early era of programming, people acknowledged the importance of the humanities in software programming/development [65] and comparisons have been actively drawn between programming and art, following the intuition and the metaphors of Knuth, and, partially, of Wallace [66–69]. These early works focused on emphasizing the aesthetic and creative nature of computer applications, while attempting to undermine the idea that coding was purely and merely a technical endeavor.

In recent years, many researchers have gone further and they have not only emphasized that programming is a highly creative process, which partly escapes a mathematical formalization and has an important, intrinsic aesthetic or even artistic flavor, but have also found several similarities in the process of programming and those characterizing various artistic or social disciplines [70–77].

1.3. Motivations, aims, and goals of the present work

Yet, as we discovered – at the time of writing at least – there are no works that have surveyed and systematized such results. We have selected a Systematic Literature Review (SLR) as an effective way to perform such survey and systematization. According to Fink (2019) [78], the SLR can be defined as "a systematic, explicit, and reproducible method for identifying, evaluating, and synthesizing the existing body of completed and recorded work produced by researchers, scholars, and practitioners". A SLR then provides a comprehensive and critical analysis of all the research conducted in the field, while synthesizing that information in productive ways, so as to offer a complete interpretation of research results, which can be beneficial for a larger audience. The importance of SLRs to scientific research lie into their analytical feature; that is, in the comprehensive and systematic methodology used to perform them [79].

The field of our research (the intersection between art and programming) is now expanding quickly, becoming ever more interdisciplinary. We therefore felt the need of systematizing and making easily available to the research community a portion of the most significant research carried out to date, focusing specifically on storytelling, as it would be too difficult and demanding to consider the whole spectrum of artistic disciplines recently related to software development.

In synthesis, our work focuses on the process of storytelling and has two aims:

- **A_1:** to inform software developers of the opportunities and the consequences of viewing software development as a creative process to a certain extent analogous to storytelling;
- **A_2:** to help researchers access easily the existing works done by others in this area.

With these aims in mind, we organized the paper as follows. Section 2 describes the methodology we adopted in conducting this research and the related study protocol. Section 3 presents and illustrates our findings, while answering our research questions. Section 4 discusses and contextualizes the results, explaining their significance and relevance for the field. Section 5 critically assesses our results, evaluating potential biases as well as potential threats to validity underlying this study. Section 6 draws some general conclusions about the difficulty and relevance of this work, while pointing out future research directions. The Appendix contains two tables: the first table (our literature log) lists all the papers we have included and analyzed for this systematic literature review; the second table describes how our work complied with the PRISMA checklist, which is further explained in Section 2.1.

2. Research methodology

We begin this section by reassuring our readers about the consistency of our thematic analysis. All the steps involved in the research protocol – and especially in the methodology section – have been carefully and collaboratively cross-checked by all members involved in this study. This was done to maximize objectivity and minimize errors, as well as the presence of potential biases.

As we mentioned in the Introduction, storytelling in software engineering has been studied from a variety of standpoints, for instance in requirements elicitation [59], in software design [77], in agile coding [80], in user experience evaluation [81], in software maintenance [82] in educating developers [83], etc. These are very different activities, which require different methods. Consequently, the papers studying such activities have applied a variety of scientific approaches and it is not easy to present to our readers a comprehensive view of such approaches. Nevertheless, we think that the relevance of the subject makes such comprehensive view of paramount importance for the future progress of our discipline.

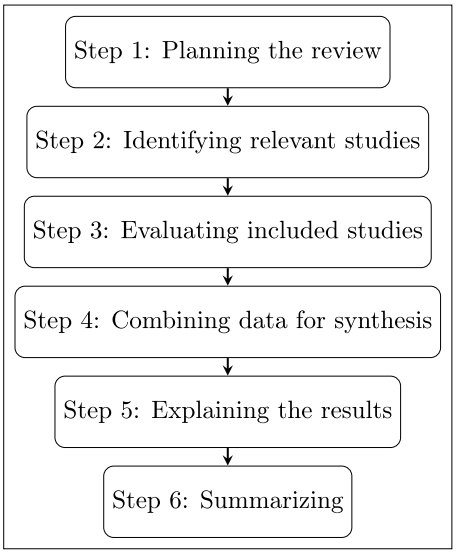

Fig. 1. Steps involved in a systematic literature review.

Systematic literature reviews, as noted above, are important tools for scientific progress and are also one of the preferred methods used by researchers to investigate in depth the state of the art of a particular research topic, as the baseline for the followup progress. As written in the Introduction, our work is a systematic literature review on the subject aimed at providing the mentioned needed review and systematization of the existing works.

Let us emphasize that the rationale and benefit for using a systematic literature review in our specific research domain lies in the unbiased examination of the potential application of storytelling to software development [84].

In our work we have strictly followed the steps highlighted in the PIECES framework [85], which are summarized in Fig. 1.

As shown in the figure, the acronym 'PIECES' stands for:

- P: Planning — we laid down the groundwork for this research by specifying the research questions guiding it. We also formulated a set of inclusion and exclusion criteria, and composed our search queries using the notorious PICO framework as a point of Ref. [86,87].
- I: Identifying — we searched a number of different databases to maximize coverage of potentially interesting works, recording overall results and selecting papers that satisfied the criteria for inclusion we outlined in our search protocol.
- E: Evaluating — we collected, read, and analyzed papers, excluding inappropriate ones.
- C: Combining — we extracted the relevant information relative to our research questions from the papers we collected.
- E: Explaining — we conducted a careful analysis of our literature log, hence providing a critical synthesis of our work. In addition, we also estimated shortcomings and limitations of our research.
- S: Summarizing — we formalized our findings and conclusions.

The PIECES model helps in establishing a generic workflow for performing research over several corpora including scientific literature. Besides illustrating the workflow; however, we also needed a detailed checklist of activities and expected outcomes for our work. This is why we used the PRISMA checklist.

2.1. PRISMA checklist

In order to report our findings in an acceptable form, we conducted a systematic literature review in accordance with the suggestions outlined by Brereton et al. (2007) and Kitchenham et al. (2009) [88,89]. See also [90–92]. The review was conducted in conformity with the PRISMA checklist [93]. Table 3 in the Appendix shows our checklist and how we used it in this study. The guideline *Preferred Reporting Items for Systematic reviews and Meta-Analyses* (PRISMA[4]) provides a guidance for the organization of a SLR, reporting study protocol, performed work, and describing research findings.

2.2. Research questions

We first defined the general goal of our research by using the Goal Question Metric model, which requires to specify the *purpose* of measurement, the *objects* to be measured, the *issues* to be measured, and the *viewpoints* from which the measures are taken [94]. In our case:

Purpose: literature analysis;

Object: scholarly papers studying software development;

Issue: application of storytelling principles to software making;

Viewpoint: software engineers and software engineering researchers.

Having set the general goal of our research in light of the above-mentioned model, we then formulated our research questions:

RQ$_1$: What are the similarities between the practice of software development and the practice of storytelling?

RQ$_2$: What are the specific goals of storytelling, as perceived by software engineers, during software development?

RQ$_3$: What are the principles underlying storytelling that can help in refining current practices in software development?

The motivation for RQ$_1$ was to gain a deep understanding with respect to what the existing literature has discovered in terms of relating these two -*seemingly*- different processes; namely software development and storytelling. In particular, we were interested in finding out the analogies that exist between such practices as well as in studying how such analogies could be used to link these practices more directly, for instance, for the purpose of educating software developers. Providing a well-thought answer to this research question is of paramount importance because it should enable researchers to realize that software development is not purely a technical field, fully formalized through logic and mathematics [95]; but rather, that it can indeed be deeply inspired, and shaped and even guided by creative activities such as those involved in storytelling.

The motivation for RQ$_2$ was to find out what general purposes or principles can be extracted from the practice of storytelling that can be generally relevant to software development. An answer to this research question promised to give an overall sense of the benefits that storytelling can have or bring about in software development.

The motivation for RQ$_3$ was to dive even further into the idea specified in RQ$_2$ and thus look for specific, essential principles of storytelling that can be practically used in software development. An answer to this third research questions promises to specify the answer given to RQ$_2$ and thus help us in formulating specific proposals for improving the current practice of software development.

2.3. Search strategy

In order to individuate relevant papers in the field, we conducted a manual search through different databases. A manual search was preferred over an automated one for two reasons, mostly: i. manually checking each entry ensures more comprehensive as well as more systematic results; ii. such a process allows a more targeted search by estimating or gauging the relevance of papers in light of their significance to the specific research questions proposed and not just to the subjects studied (software development and literature) that in this case were quite broad.

2.4. Search sources and query construction

We used four repositories to perform our searches. The repositories we searched are:

- ACM Digital Library
- Microsoft Academic
- Google Scholar
- Research Gate

We performed an initial search in the repositories mentioned above by using specific keywords. However, this search method proved to be rather ineffective, as it resulted in the inclusion of a very large amount of unrelated papers. Thus, to sort out irrelevant sources, we formulated specific search queries, which targeted papers that more directly fell into our research interest. We applied these queries to searches performed on two repositories (ACM DL and Microsoft Academic), exploiting their functions for advanced searches. Instead, for searches in Google Scholar and Research Gate we manually searched for relevant keywords without producing specific search queries. This is an acceptable practice for inclusion of relevant papers in a review, according to Petersen et al. (2015) [96].

Queries used for searching ACM DL and Microsoft Academic:

Query 1

("software design" **OR** "software development" **OR** "software system")
AND
(storytelling **OR** story **OR** "story-based approach" **OR** narrative **OR** metaphor)

Query 2

(programming **OR** debugging **OR** "requirements elicitation")
AND
(storytelling **OR** (writing **AND** similarity))

[4] http://www.prisma-statement.org, 2020.

Query 3
storytelling (architecture **OR** pattern) for software (design **OR** development)

Query 4
storytelling (similarities **OR** analogies) with software (design **OR** development)

Query 5
what (storytelling **OR** narrative) can do for software (design **OR** development)

2.5. Inclusion and exclusion criteria

Having specified a number of search queries, which allowed us to narrow down the focus of our work, we then formulated a set of inclusion and exclusion criteria. These criteria helped identifying the most relevant studies among those gathered through our searches and allowed us to filter out improper or irrelevant sources. The formulation of inclusion/exclusion criteria is a necessary step for any systematic literature review. Inclusion criteria typically define a set of characteristics for papers to be included in a given study. Exclusion criteria, instead, determine a set of characteristics, which prevent answering the research questions formulated or result in unfavorable findings [97]. We consistently applied such criteria, while scanning papers obtained through search queries and manual searches. This allowed us to record and log only those sources we deemed appropriate for further reading. A paper was included into our literature log only if it satisfied each inclusion criteria. On the contrary, a paper was disregarded if it satisfied any exclusion criterion.

Inclusion criteria:

- Paper is written in English.
- Paper is peer reviewed.
- The reported outcomes are analogies drawn between storytelling and software development, and includes similarities of processes, practices or resulting products.
- Paper is focused on practical application of storytelling principles for software development.
- In this systematic literature review we considered papers published between 1994 and 2020. The time span is ample but the reason for this choice is simple: we aimed to cover all possible publications that could be useful to answer our research questions. The starting year was decided after checking which papers or books we could find on the topic of storytelling in software engineering, that is not a traditional topic of research. The first paper we found is [98], so we started from its year of publication.

Exclusion criteria:

- Editorials, reviews, opinion pieces were excluded as such works, we believe, do not provide sufficient level of reliability and focus.
- Papers considering storytelling in the general context of the design activity were excluded
- Paper analyzing other formats of storytelling, aside from textual one were also excluded.

2.6. Search process

Searching through ACM DL:

1. In total, applying the queries we got 340 papers before the application of the exclusion criteria.
2. All papers were written in English without duplicates.
3. All papers were "archival publications" in scientific journals. As a consequence, neither grey literature nor non-peer reviewed sources appeared in the search.
4. 7 papers were excluded in light of the exclusion criterion referring to the general design context.
5. 12 papers were excluded because of inappropriateness of the storytelling format considered.
6. 293 papers were eliminated because they violated the third and/or the fourth inclusion criteria.
7. 15 more papers were excluded due to insufficient amount of information provided.
8. As a result, 13 papers were included into the log for further analysis.

Searching through Microsoft Academic:

1. In total, applying the queries we got 698 papers before the application of the exclusion criteria.
2. After constraining the resulting records to comply to the criteria specified in Section 2.5, the final count was 70 papers.
3. All of the results were papers written in English without duplicates. All these papers were also "archival publications" in scientific journals. This allowed us to eliminate grey literature and non-peer reviewed sources.
4. 15 papers were excluded as they met the second exclusion criteria.
5. Finally 40 papers were also removed due to the fact that the content of their research did not relate to our third and fourth inclusion criteria.
6. 15 papers were accepted and further added to our log.

Papers selected through Google Scholar and Research Gate:

1. Papers selected with the help of these two repositories were not searched by using any of the search queries specified above; rather, they were included manually, which is the second most common way of including references in a review according to Petersen et al. (2015) [96].
2. At the end of the manual search on these two additional databases, 16 papers were retrieved from Google Scholar and 7 papers gathered from Research Gate. 23 papers were included in our log following this procedure.

The PRISMA flow diagram shown in Fig. 2 depicts this process visually for the reader.

2.7. Selection process

At the end of the search process, we gathered a variety of papers, with different levels of technical content. In order to accept the papers as a contribution to our log, thus informing our systematic literature review, we had to evaluate them using the exclusion and inclusion criteria mentioned in Section 2.5 above. As noted, a paper was approved if it met the inclusion criteria listed and discarded – on the contrary – if it met any of the exclusion criteria or failed to meet the first inclusion criteria. This process was carried out manually by the researchers involved in this study, without the aid of any automatic tool. This allowed us to cross-check the papers included. As a result of this process, we selected 51 for our review (they are listed in Appendix, Table 2).

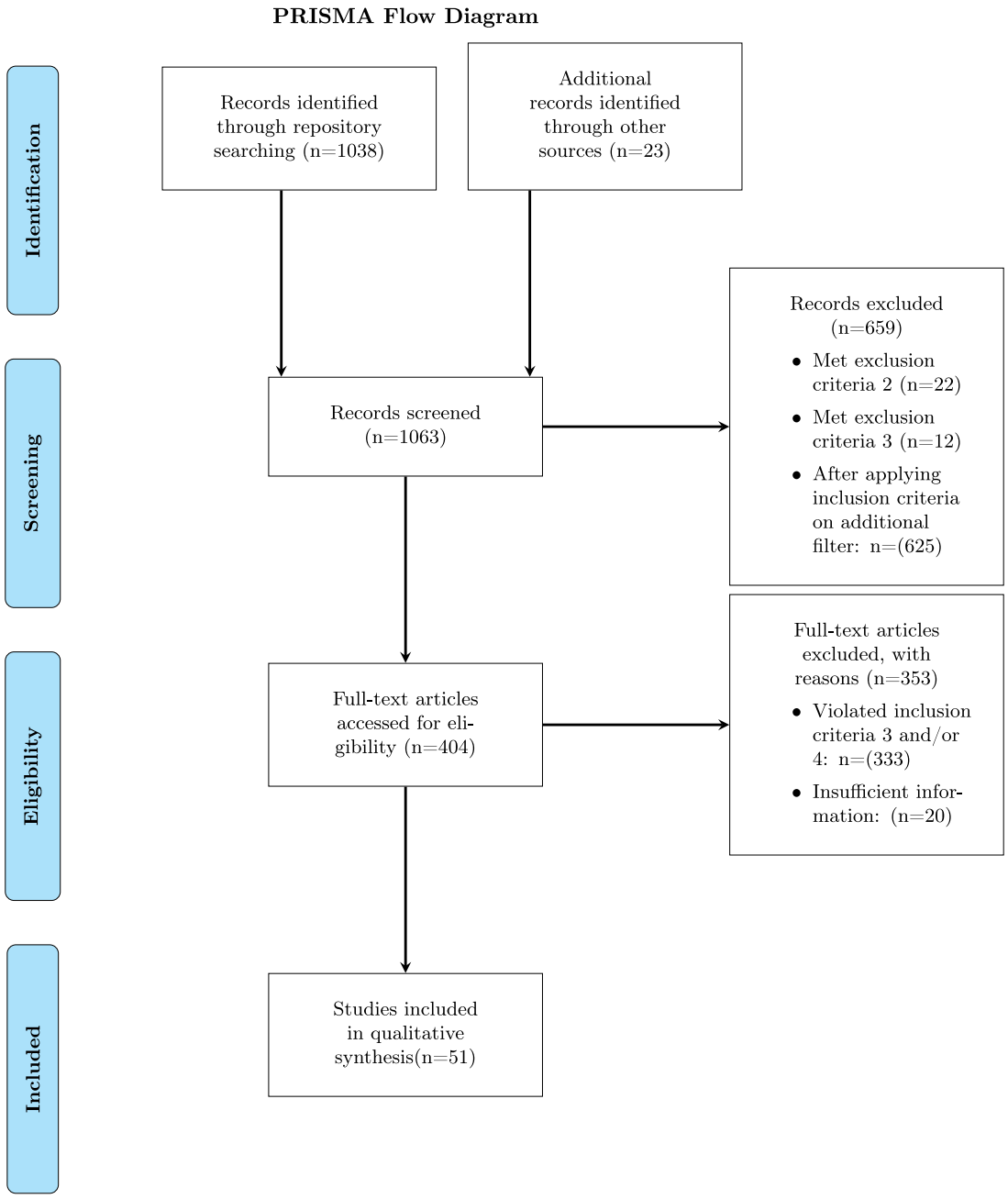

Fig. 2. Prisma flow diagram.

2.8. Data collection process

Having specified the list of papers included in the log, we then started extracting potentially useful data from them. For this reason, we clustered the papers and read them very thoroughly. For each paper, the researcher who read it collected the most relevant information. Such information was then stored in the reading log. The purpose of this process, which mostly involved collecting notes, was to allow us to retrieve and readily access previously researched data and information, without intensively re-reading the relevant papers upon need. The log was reviewed and cross-checked by other authors, and – based on constructive feedback received – it was further modified and improved to reduce the degree of subjectivity and increase instead the degree of objectivity and rationality.

Data extracted from papers' metadata, namely *authors*, *title*, *keywords*, *year*, and *affiliations* were used as identification information, while data from the paper and related notes contained information valuable for answering the research questions. In case one paper addressed multiple research questions, the data was organized in a separate cell, one for each Research Question.

2.9. Data analysis

The data included into the reading log was then further examined. The following problem quickly emerged. The vast majority of the sources collected were based on qualitative methods of analysis. However, the studies included were not homogeneous in terms of methodology and, sometime, had different structure. In such cases, meta-analysis might be inappropriate for data synthesis [99]. Therefore, we decided to manually analyze the

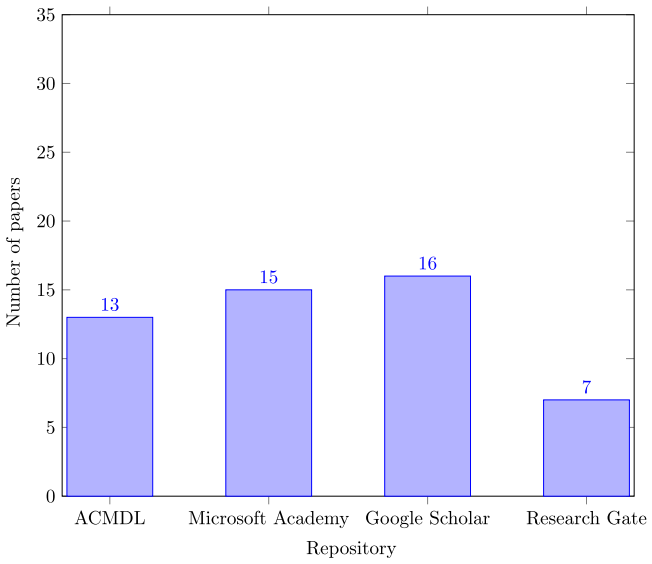

Fig. 3. Distribution of papers across the four repositories we used.

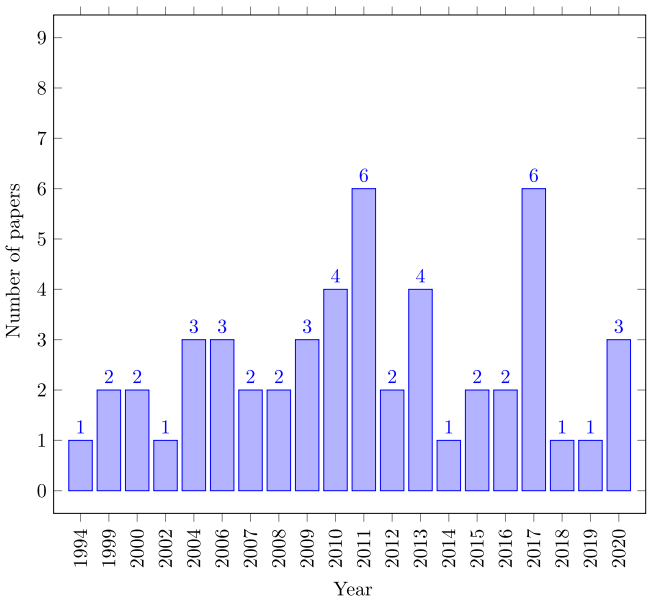

Fig. 4. Distribution of papers by year of publication.

papers and to perform a "textual narrative synthesis" [100]. In other words, we classified the papers in homogeneous groups. The synthesis process, including the steps characterizing it, is described below:

1. study the relations between the research question and the corresponding papers;
2. definition of a criterion through which one could group and cluster and classify papers;
3. production of specific commentaries for each paper within a cluster or a subgroup;
4. synthesis of findings for each subgroup or subcategory;
5. formulation of an overall conclusion informing the research question.

3. Results

3.1. Data collected

Fig. 3 shows the distribution across repositories of papers collected.

Fig. 4 shows the distribution of the papers by year of publication. We considered the interval between 1994 and 2020. We did not observe a significant increase or a significant decrease in published papers on these topic over the years. The number of papers gathered remained in fact rather stable, except for 2011 and 2017, where a mild increase was observed. We note that most of the existing works focused on the Requirements Engineering part of the Software Development Process.

Fig. 5 shows which areas of software development have been covered by the studies we selected. The authors decided to include this information in the review because by using this graph, one could deduct which areas have not been covered yet, and this could potentially assist other researchers in selecting the most ripe areas for exploration.

Having clustered papers in such a way, we then looked at how data from the papers selected could help us answering our research questions.

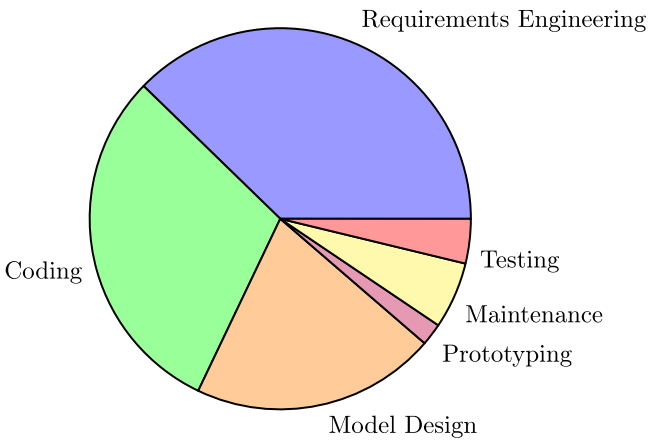

Fig. 5. Distribution of papers per specific software engineering phase.

3.2. RQ1: What are the similarities between the practice of software development and the practice of storytelling?

As mentioned above, we performed a Textual Narrative Synthesis for this research question by grouping relevant papers into two main subcategories. These were:

- papers describing processes of storytelling and software development — two papers;
- papers describing processes of storytelling and production of software artifacts — four papers.

Each subcategory or cluster considered similarities between various aspects of the software engineering process and the literature domain.

Findings. More specifically, papers from the first subcategory typically explained or described what stages and steps are involved in these activities, highlighting commonalities in nature and character. Even though different researchers use different names for certain stages and degrees of granularity for the classification into specific steps, the essence and order of activities remain the same.

In this paragraph, we will use the term "project" interchangeably to refer to software and literary pieces of work. So, a project starts out with a *plan*, which specifies the actions needed in order to fulfill the project's vision.

The second step involved is then to set clear objectives, define priorities, and a set of necessary tools that can be used to collect relevant information about the project's topic. After that, one should structure collected data, process it, and select relevant pieces of information.

The next step involved the translation of high-level ideas into corresponding representations in a natural or programming language. At the end of the process, there should be a review session to reflect upon the results, estimate their quality and conformance with the initial requirements, and make amendments if needed [56,101].

Those described in the previous paragraph were the most important commonalities in actions observed in storytelling and software development.

What can we say about the possible resemblance of the artifacts produced during such activities? The second subcategory of papers included in the review was concerned with this specific question.

As we mentioned earlier, both writers and software developers need to carefully plan their work. Andreas Munk-Madsen and Peter Bøgh Andersen [57] decided to examine how elements of engaging stories can be applied in writing plans for software projects and whether this idea can bring about any benefit for the discipline. They singled out three important elements:

- *a conflict*, describing threats to the project;
- *the presence of actors* played by developers and customers;
- *a plot*, which is a project plan represented in a narrative form.

The researchers concluded that project managers might indeed benefit from applying the storytelling perspective during planning activities in software development.

Another important stage in software development involves the collection of relevant information for the project by using requirements elicitation practice. According to Naoufel Boulila et al. [59], an approach based on storytelling is far more effective than brainstorming for gathering the requirements and capturing the tacit knowledge. Work by Rubia Fatim et al. [60], also showed positive statistical results underlying the adoption of a "story elements-based elicitation" method. In particular, both these papers mentioned the following elements of storytelling as applicable and beneficial for software developers:

- *a conflict* – which is basically the problem to solve by the composed requirement;
- *a set of characters* – any entities playing a role in the development of software and elicitation of requirements (users, stakeholders, the system itself);
- *an environment* – the time and the place in which the story is taking pace as well as the general context of the problem, which includes information about technological and business environments;
- *a proposed resolution or a main concept underlying the resolution* – this can be understood as a project goal.

Overall, one can say that the main advantage of adopting an approach based on storytelling lies in the specification of more detailed and complete requirements. This is due to the transfer of tacit knowledge as well as to an increased engagement of developers.

There are some similarities between the practice of storytelling and the process of coding a program, as shown by Jones et al. (2011) [61]. The researchers involved in this study observed the presence of many analogies between storytelling and components of an Alice programming environment [102]. Subsequently, they generalized the case up to the mapping between elements of the English grammar and elements of a computer program [61].

Table 1
Elements of storytelling vs elements of coding [61].

English grammar	"Program's" grammar
Nouns	Classes and objects
Adjectives	Class fields and properties
Verbs	Methods and behaviors
Sentences	Code instructions
Paragraphs	Methods

Some analogies between the two processes are summarized in the Table 1.

3.3. RQ2: What are the specific goals of storytelling, as perceived by software engineers, during software development?

To examine which goals developers aim at while applying storytelling techniques in software engineering, we clustered the papers we selected into four main categories, describing the most frequently and generally accounted goals:

- to capture and transfer tacit knowledge — four papers;
- to facilitate comprehension — four papers;
- to facilitate communication — four papers;
- to bring in emotional expressiveness and creativity — three papers.

We would like to note that the categories above-mentioned are not necessarily mutually exclusive. Thus, some papers potentially belong to more than one category.

Findings. As the amount of data available grows, there is an increasing need to transform and systematize this information into reliable knowledge [103]. This is crucially important for the field of software engineering, where developers need to document and formalize tacit knowledge at a great speed. To cope with this need problem, a proposed solution was to "externalize" or to convert tacit knowledge into explicit one [103,104]. In particular, our review highlighted some key benefits of adopting a story-driven approach to software development:

- assistance in constructing a shared metaphor;
- visibility of the project's progress;
- easiness of referencing to the current status of the project;
- discovery of possible issues.

In particular, it was observed that storytelling can be profitably applied in requirements elicitation and documentation of detailed non-functional requirements [58,59].

Our review also highlighted another benefit of using storytelling principles in software development. Such principle enhance the comprehension of system's design among developers. Since parties involved in the development process have knowledge and skills from very different domains (e.g. marketing, engineering, design, management), it is sometimes difficult to expect from them the same level of understanding about the project [141]. To facilitate this process one the suggested solutions was the adoption of the so-called Experiential Design Language [131]. This method uses "appealing abstractions" and illustrates the project from different perspectives, taking into account the different roles of the people involved in it. It was observed that creating metaphors is another literature technique, which can highly beneficial for software development. In particular, such a technique can help developers by:

Table 2
Papers included in final log, listed chronologically.

No.	Author	Year	Title	Source	Ref.
1.	Hasse Clausen	1994	Designing computer systems from a human perspective: The use of narratives	Google scholar	[98]
2.	Line Dubé, Daniel Robey	1999	Software stories: three cultural perspectives on the organizational practices of software development	Google scholar	[105]
3.	David Snowden	1999	Story telling: an old skill in a new context	Microsoft academic	[106]
4.	Peter Lloyd	2000	Storytelling and the development of discourse in the engineering design process	Google scholar	[107]
5.	Stefana Broadbent, Francesco Cara	2000	A narrative approach to user requirements for web design	Microsoft academic	[108]
6.	Dan Gruen, Thyra Rauch, Sarah Redpath, Stefan Ruettinger	2002	The use of stories in user experience design	ResearchGate	[109]
7.	Raphael Perret, Marcos Borges, Flávia Maria Santoro	2004	Applying group storytelling in knowledge management	ACMDL	[103]
8.	Mike Cohn	2004	User Stories Applied: For agile software development	Microsoft academic	[110]
9.	Tammy VanDeGrift	2004	Coupling pair programming and writing: learning about students' perceptions and processes	Google scholar	[111]
10.	M.R.K. Krishna Rao	2006	Storytelling and puzzles in a software engineering course	Google scholar	[83]
11.	Andreas Munk-Madsen, Peter Bøgh Andersen	2006	Storytelling in projects: Transforming project plans into stories	Google scholar	[57]
12.	Brian Bussell, Stephen Taylor	2006	Software development as a collaborative writing project	ACMDL	[112]
13.	Adriana Cristina de Oliveira, Renata Mendes de Araujo, Marcos Borges	2007	Telling stories about system use: Capturing collective tacit knowledge for system maintenance	Google scholar	[113]
14.	James Siddle	2011	Choose your own architecture: Interactive pattern storytelling	Microsoft academic	[114]
15.	Sean Hammond, Helen Pain, Tim J. Smith	2008	Children's story authoring with Propp's morphology: An exploratory study	Google scholar	[115]
16.	Roger McDermott, Gordon Eccleston, Garry Brindley	2008	More than a good story — can you really teach programming through storytelling?	Microsoft academic	[116]
17.	Stephen Hayne	2009	Using storytelling to enhance information systems knowledge transfer	Google scholar	[117]
18.	Erik Wende, Parissa Haghirian	2009	Storytelling as a tool for knowledge transfer in the IT industry	Google scholar	[118]
19.	Viviane Laporti, Marcos Borges, Vanessa Braganholo	2009	Athena: A collaborative approach to requirements elicitation	Microsoft academic	[119]
20.	Sabine Madsen, Lene Nielsen	2010	Exploring persona-scenarios - Using storytelling to create design ideas	Google scholar	[120]
21.	Debra Richardson, Ban Al-Ani, Hadar Ziv	2010	Requirements engineering at the margins: avoiding technological hubris through alternative approaches	Microsoft Academic	[121]
22.	Whitney Quesenbery, Kevin Brooks	2010	Storytelling for user experience: Crafting stories for better design	Microsoft academic	[81]
23.	Quinn Burke, Yasmin B. Kafai	2010	Programming & storytelling: opportunities for learning about coding & composition	Microsoft academic	[122]
24.	Dr. Naoufel Boulila, Anne Hoffmann, Dr. Andrea Herrmann	2011	Using storytelling to record requirements: Elements for an effective requirements elicitation approach	ResearchGate	[59]
25.	David West, Jenny Quillien	2011	Patterns for story craft	ACMDL	[5]
26.	Hamizah Mohamad Hariri, Marini Abu Bakar, Abdullah Mohd Zin	2011	Story telling approach for integrating software blocks	ACMDL	[123]
27.	Mary Elizabeth Jones, Melanie Kisthardt, Marie Cooper	2011	Interdisciplinary teaching: Introductory programming via creative writing	ACMDL	[61]

(*continued on next page*)

- providing a common vocabulary for all members of the development team;
- having additional diagrams, visuals and explanations;
- conveying the rationale behind high-level architectural decisions;
- depicting different system view's and
- describing perspectives of a particular side involved in the project (e.g., stakeholders/managers/engineers/designers);

In general, it was observed that metaphors improved the team's overall understanding of quality attributes and prevented incorrect "low-level design decisions" from occurring. Stories also provided design rationales, architectural details and system's views without the need of lengthy and complicated documentation [141].

As we already mentioned elsewhere and above, if one wants to get valuable results in software development, it is of paramount importance to allow for a variety of positions and roles to be taken in the production of a software, as well as to establish reliant, timely and pervasive process of communication between

Table 2 (*continued*).

28.	Roman Knöll, Vaidas Gasiunas, Mira Mezini	2011	Naturalistic types	ACMDL	[124]
29.	Michael Keeling, Michail Velichansky	2011	Making metaphors that matter	ACMDL	[125]
30.	Madeleine Kusoffsky	2012	Leveraging storytelling in visual analytics by redesigning the user interface	Microsoft academic	[126]
31.	Quinn Burke	2012	The markings of a new pencil: Introducing programming-as-writing in the middle school classroom	Google scholar	[127]
32.	Philip van Allen, Hye Mi Kim, Joshua McVeigh-Schultz, Daniel Lara, Brooklyn Brown	2013	AniThings: Animism and heterogeneous multiplicity	ACMDL	[128]
33.	Arjen Uittenbogaard	2013	Storytelling for software professionals	Google scholar	[129]
34.	Erik Wende, Gregory King, Gerhard Schwabe	2013	Exploring storytelling as a knowledge transfer technique in offshore outsourcing	Microsoft academic	[130]
35.	Nik Boyd	2013	Software metaphors	ACMDL	[62]
36.	Azad M.Madni, Marcus Nance, Michael Richey, William Hubbard, Leroy Hanneman	2014	Toward an experiential design language: Augmenting model-based systems engineering with technical storytelling in virtual worlds	ResearchGate	[131]
37.	Angelo Gaeta, Matteo Gaeta, Giuseppe Guarino, Sergio Miranda	2015	A smart methodology to improve the story-building process	Google scholar	[132]
38.	Manuel Brhel, Hendrik Meth, Alexander Maedche, Karl Werder	2015	Exploring principles of user-centered agile software development	Microsoft academic	[133]
39.	Kimberly A. Gausepohl, Woodrow W. Winchester,Tonya L. Smith-Jackson, Brian M. Kleiner, James D. Arthur	2016	A conceptual model for the role of storytelling in design: leveraging narrative inquiry in User-Centered Design (UCD)	Micorsoft academic	[134]
40.	Murat Yilmaz, Berke Atasoy, Rory V. O'Connor, Jean-Bernard Martens, Paul Clarke	2016	A story-driven approach to support software practitioners	ACMDL	[104]
41.	Felienne Hermans and Marlies Aldewereldb	2017	Programming is writing is programming	ACMDL	[56]
42.	Kevin Devaney, James Johnson	2017	Storytelling as a key enabler for systems engineering	ResearchGate	[58]
43.	Raffaele Ciriello, Alexander Richter, Gerhard Schwabe	2017	When prototyping meets storytelling	ResearchGate	[8]
44.	Nguyen Thien Khanh, Jirapun Daengdej, Habibi Husain Arifin	2017	Human stories - A new written technique in agile software requirements	ACMDL	[135]
45.	Edhy Rustan	2017	Learning creative writing model based on neurolinguistic programming	Microsoft academic	[136]
46.	Zahra Shakeri Hossein Abad, Alex Shymka, Jenny Le, Noor Hammad, Guenther Ruhe	2017	A visual narrative path from switching to resuming a requirements engineering task	Microsoft academic	[137]
47.	Chao Tong, Richard Roberts, Rita Borgo, Sean Walton, Robert S. Laramee, Kodzo Wegba, Aidong Lu, Yun Wang, Huamin Qu, Qiong Luo and Xiaojuan Ma	2018	Storytelling and visualization: An extended survey	Google scholar	[138]
48.	Rubia Fatima, Affan Yasin, Lin Liu, Jianmin Wang, Wasif Afzal, Atif Yasin	2019	Improving software requirements reasoning by novices: a story-based approach	ResearchGate	[60]
49.	Paolo Ciancarini, Sergey Masyagin, Giancarlo Succi	2020	Software design as story telling: reflecting on the work of Italo Calvino	ACMDL	[77]
50.	Ziva Hassenfeld, Marina Bers	2020	Debugging the writing process: Lessons from a comparison of students' coding and writing practices	ResearchGate	[101]
51.	Rodi Jolak, Maxime Savary-Leblanc, Manuela Dalibor, Andreas Wortmann, Regina Hebig, Juraj Vincur, Ivan Polasek, Xavier Le Pallec, Sebastien Gerard, Michel R. V. Chaudron	2020	Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication	Google scholar	[139]

and among team's members [58,142,143]. One of the possible activities where storytelling is applied to successfully establish communication patterns between developers and customers is the process of collection and analysis of users' experiences [144]. The major benefits obtained by adopting such an approach are described below. Specifically, they involve:

- obtaining more sincere and detailed feedback by helping customers to concentrate on their emotions and experience;
- keeping the focus on customers' values;
- communicating about the project with people;
- using these patterns for promotional campaigns, marketing as well as for presentations to potential customers.

In other words, it is demanded to produce helpful narratives, which can – on the one hand – allow stakeholders deciding upon the system's needs; and – on the other hand – help software engineers understanding and assessing the system's feasibility [145].

Last but not least, a proclivity to enrich the process of software design with emotional units and occurrences of creativity is observed. Storytelling helps to turn a formal process and relatively standardized process (such as that of software design) into an engaging activity realized by means of visual representations

Table 3
PRISMA 2020 Checklist Template taken from: [140].

Section and Topic	Item #	Checklist item	Location where item is reported
Title			
Title	1	Identify the report as a systematic review.	1
Abstract			
Abstract	2	See the PRISMA 2020 for Abstracts checklist	2.1
Introduction			
Rationale	3	Describe the rationale for the review in the context of existing knowledge.	1
Objectives	4	Provide an explicit statement of the objective(s) or question(s) the review addresses.	1
Methods			
Eligibility criteria	5	Specify the inclusion and exclusion criteria for the review and how studies were grouped for the syntheses.	2.5
Information sources	6	Specify all databases, registers, websites, organizations, reference lists and other sources searched or consulted to identify studies. Specify the date when each source was last searched or consulted.	2.4
Search strategy	7	Present the full search strategies for all databases, registers and websites, including any filters and limits used.	2.3
Selection process	8	Specify the methods used to decide whether a study met the inclusion criteria of the review, including how many reviewers screened each record and each report retrieved, whether they worked independently, and if applicable, details of automation tools used in the process.	2.7
Data collection process	9	Specify the methods used to collect data from reports, including how many reviewers collected data from each report, whether they worked independently, any processes for obtaining or confirming data from study investigators, and if applicable, details of automation tools used in the process.	2.6
Data items	10a	List and define all outcomes for which data were sought. Specify whether all results that were compatible with each outcome domain in each study were sought (e.g. for all measures, time points, analyses), and if not, the methods used to decide which results to collect.	2.8
	10b	List and define all other variables for which data were sought (e.g. participant and intervention characteristics, funding sources). Describe any assumptions made about any missing or unclear information.	2.8, 2.9
Study risk of bias assessment	11	Specify the methods used to assess risk of bias in the included studies, including details of the tool(s) used, how many reviewers assessed each study and whether they worked independently, and if applicable, details of automation tools used in the process.	5.3
Effect measure	12	Specify for each outcome the effect measure(s) (e.g. risk ratio, mean difference) used in the synthesis or presentation of results.	–
Synthesis methods	13a	Describe the processes used to decide which studies were eligible for each synthesis (e.g. tabulating the study intervention characteristics and comparing against the planned groups for each synthesis (item #5)).	2.9
	13b	Describe any methods required to prepare the data for presentation or synthesis, such as handling of missing summary statistics, or data conversions.	–
	13c	Describe any methods used to tabulate or visually display results of individual studies and syntheses.	3.1
	13d	Describe any methods used to synthesize results and provide a rationale for the choice(s). If meta-analysis was performed, describe the model(s), method(s) to identify the presence and extent of statistical heterogeneity, and software package(s) used.	2.9
	13e	Describe any methods used to explore possible causes of heterogeneity among study results (e.g. subgroup analysis, meta-regression).	–
	13f	Describe any sensitivity analyses conducted to assess robustness of the synthesized results.	–
Reporting bias assessment	14	Describe any methods used to assess risk of bias due to missing results in a synthesis (arising from reporting biases).	5.2
Certainty assessment	15	Describe any methods used to assess certainty (or confidence) in the body of evidence for an outcome.	5.3

(*continued on next page*)

that stimulate creative thinking and emotional involvement of all team members [104]. Moreover, it is shown that shared stories and group work help developing a sense of responsibility among team members, which ultimately enhances commitment and dedication to the project [103,104]. In addition, an approach grounded on storytelling is paramount for capturing and transferring emotional aspects of software artifacts (such as user requirements). Storytelling about users make software requirements more compelling, causes more empathy, and ultimately facilitates a full understanding of customers' needs [135].

Table 3 (*continued*).

Results			
Study selection	16a	Describe the results of the search and selection process, from the number of records identified in the search to the number of studies included in the review, ideally using a flow diagram.	2.6
	16b	Cite studies that might appear to meet the inclusion criteria, but which were excluded, and explain why they were excluded.	2.6
Study characteristics	17	Cite each included study and present its characteristics.	–
Risk of bias in studies	18	Present assessments of risk of bias for each included study.	5
Results of individual studies	19	For all outcomes, present, for each study: (a) summary statistics for each group (where appropriate) and (b) an effect estimate and its precision (e.g. confidence/credible interval), ideally using structured tables or plots.	4
Results of syntheses	20a	For each synthesis, briefly summarize the characteristics and risk of bias among contributing studies.	4
	20b	Present results of all statistical syntheses conducted. If meta-analysis was done, present for each the summary estimate and its precision (e.g. confidence/credible interval) and measures of statistical heterogeneity. If comparing groups, describe the direction of the effect.	–
	20c	Present results of all investigations of possible causes of heterogeneity among study results.	–
	20d	Present results of all sensitivity analyses conducted to assess the robustness of the synthesized results.	–
Reporting biases	21	Present assessments of risk of bias due to missing results (arising from reporting biases) for each synthesis assessed.	5.2
Certainty of evidence	22	Present assessments of certainty (or confidence) in the body of evidence for each outcome assessed.	5.3
Discussion			
Discussion	23a	Provide a general interpretation of the results in the context of other evidence.	
	23b	Discuss any limitations of the evidence included in the review.	5.1
	23c	Discuss any limitations of the review processes used.	5.1
	23d	Discuss implications of the results for practice, policy, and future research.	4
Other information			
Registration and protocol	24a	Provide registration information for the review, including register name and registration number, or state that the review was not registered.	–
	24b	Indicate where the review protocol can be accessed, or state that a protocol was not prepared.	2.1
	24c	Describe and explain any amendments to information provided at registration or in the protocol.	–
Support	25	Describe sources of financial or non-financial support for the review, and the role of the funders or sponsors in the review.	–
Competing interests	26	Declare any competing interests of review authors.	7
Availability of data, code and other materials	27	Report which of the following are publicly available and where they can be found: template data collection forms; data extracted from included studies; data used for all analyses; analytic code; any other materials used in the review.	8

3.4. RQ3: What are the principles underlying storytelling that can help in refining current practices in software development?

To answer this research question, we spit the remaining papers into four groups or categories:

- proposals to refine software development as a whole - 2 papers;
- proposals to refine the system designed - 3 papers;
- proposals to refine programming - 2 papers;
- proposals to refine requirements engineering - 3 papers.

Findings. We begun our investigation with papers that offered some methods or views for refining the process of software development. The main objective of Yilmaz et al. (2016) [104] was to estimate the efficiency of adapting methods from visual storytelling in software development processes. In particular, the researchers involved in this study pointed out that storytelling may help capturing technical information, as well as conveying contexts, which may help the team externalizing "context-specific tacit knowledge", hence boosting creative efforts. A method, called Storyply, was proposed to implement such a process. Its steps are summarized below:

- Firstly, one gathers experiences of software development and all relevant information about the project.
- Secondly, one interprets such data in terms of story elements such as actors, domains, artifacts.
- Thirdly, one identifies the main stakeholders.
- Fourthly, one represents the collected experiences visually.

The second paper we included in this category is [77]. This is essentially a reflection on analogies between software design and writing through a careful analysis of Italo Calvino's work. The aim of the paper was to suggest improvements to software development processes and practices based on literary reflections. The suggestion is that developers should borrow principles and practices from great masters not only of software engineering but also of storytelling: and Calvino was a master storyteller, worldwide known. The paper in particular noted how the application of five basic principles related to good writing (such as lightness, rapidity, precision, visibility, and multiplicity), can bring about many benefits to software developers.

The second group or category of papers looked at ways to refine the system design. In particular, the goal of Gruen et al. (2002) [109] was to demonstrate the utility of storytelling in software design while defining the elements of a persuasive and

engaging story. Researchers found out that there are several benefits of using storytelling for the design process. These benefits involve: i. the identification of users' personalities; ii. the construction of a common vision for team members; and iii. the estimation of the status of design work performed. The main contribution of the paper was the definition of the story elements needed in the design of a *compelling system*: fleshed-out characters, detailed setting, goals and obstacles, motivation, causality, dramatic elements. Keeling and Velichansky (2011) [125] studied instead some principles underlying metaphors production in the literature, which could serve as guidelines for writing productive and meaningful metaphors while developing software. These principles are:

- representing a single view;
- dealing with only one type of structure;
- giving clear guidance concerning design decisions;
- shedding light on system properties;
- drawing on a shared experience;

Finally, Clausen (1994) [98] proposed a model for using narratives in the system's design process. The model suggests that the workflow should be organized as follows:

- there should be two types of documents: the narrative for stakeholders to decide upon system's need, and drafts for engineers to assess and understand the system's feasibility;
- while formulating the problem, the designer should come up with a theme that is related to certain users. The theme will help narrowing down the domain of the design, thus concentrating only on the aspects related to those users;
- the designer should prepare a clear description of the sort of people that the system is intended to target.
- the designer should conduct a qualitative study and formulate histories from different users' perspectives;
- the designer should identify problems, analyze them carefully, and come up with a model of the situation as well as reflect on the more general question of whether it is possible to construct a system to solve the problems faced;
- the designer should produce precise drafts of the system as a whole and of its parts;
- the designer should write scenarios about future developments of the system as well as make them actual for the users.

Moving on to the category that proposed methods of refining programming, we started out by analyzing [123], which set out to design a software for integration of different blocks by using a storytelling-based approach. The Block-Based Storytelling application was implemented to allow end users designing block-based software as a series of interconnected stories. The primary result of the paper was therefore to show that the usage of textual description may help users to learn programming and facilitate implementation of logical reasoning. Building an expanding on this study, VanDeGrift (2004) [111] explored and investigated the benefits of using pair programming, coupled with individual written reports, in an introductory course in computer science. In particular, researchers found out that:

- Pair programming turned out to be valuable to the students.
- Coupled with the pair programming projects, written reports served as accountability mechanisms to make sure that both partners understood project solutions.
- The written reports were beneficial for students as they provided them with a platform to explain and better comprehend their solutions. The reports also proved to be useful as windows to gain an understanding over students' challenges.

The last group or category of papers selected focused on principles that could improve requirements engineering. Conveying needs via storytelling arguably provides more flexibility while enabling a more intuitive understanding, as shown in [59]. This paper compared the efficacy of adopting a storytelling approach with the efficacy of adopting a brainstorming approach in requirements elicitation process. Traditional approaches for requirements elicitation limit a user to particular patterns or models for the formulation of requirements. However, this research pointed out that elements of Storytelling (*Conflict, Theme, Setting, Plot, Characters and Point of view*) can be successfully used in requirements engineering.

On a similar vein, but from a different angle, Abad et al. (2017) [137] proposed a new framework to support the reasoning process underlying task switching. The proposed framework has two layers: data analysis and visualization. The framework contributes to lower the amount of cognition of requirements communication and interruptions. An experiment was setup to compare storytelling with brainstorming when requirements management is performed with frequent context switches. The result of the experiment conducted using the framework showed that the Storytelling group displayed more engagement and emotional connection between team members. They managed to produce three times more requirements than the group with brainstorming approach. As a result, there were more requirements and use cases discovered with Storytelling. In addition, the requirements produced by the storytelling group were more detailed, stated in a clearer way, and provided a broader view of the problem.

Finally, Madsen and Nielsen (2009) [120] wanted to provide a guide on how to write *persona* scenarios and how such stories should look like to be comprehensive and design-oriented. Some elements from the Narrative theory [146] were taken as a basis for elements constituting persona scenarios. The elements are *characters*, *time*, *problem*, *setting*, *opening episode*, *episodes*, *resolution*, *plot*, *overall story*, *narrator's perspective*. These elements were adapted accordingly in order to reflect the main goal of the scenarios, that is to illustrate the system's users and their motivations to use the system. As a result of the research, the following recommendations for writing scenarios were derived:

- Firstly, events should be described specifically and in details.
- Secondly, the persona's problem must be examined thoroughly and solved within the scenario.
- Finally, the focus should be kept on the persona's problem and the concern of using the designed system to solve this problem.

4. Discussion

In this section, we contextualize our findings and discuss their significance and importance for the field. We also show how our systematic literature review may bring new valuable perspectives and profitable insights to the practice of software engineers.

4.1. RQ1: What are the similarities between the practice of software development and the practice of story-telling?

There are indeed similar stages and analogous processes underlying these activities. In particular, in light of our analysis, we believe there exists a common template or model for creating pieces of literature (such as stories) and piece of software (such as programming). The following model represents the stage model of writing according to Huizenga [29], that we mentioned in the Introduction Section.

This model of storytelling can be represented as follows:

1. *Planning future work;*
2. *Setting goals;*
3. *Collecting data;*
4. *Processing and selecting data;*
5. *Translating ideas;*
6. *Reviewing results.*

Needless to say, this model can be incrementally applied to hone and perfect the project's quality over several sessions.

If storytelling and software development correspond and overlap at least in some processes, then such an overlapping we expect should also affect the artifacts or products produced through such activities. Consider as an illustration of this point, the primary, most basic elements underlying any story, which can be specified as follow – *a context, a conflict, characters, a plot, and a resolution*. Such elements are also used to write project plans, to formulate project requirements, to describe software architectures, to discuss design solutions, and last but not least to teach programming. Some storytelling can happen at different levels during a development project. For instance, design patterns are described as a vocabulary of *stories within a context, some conflicting forces, some participating entities, a discussion, and a solution* [147].

The existence of a positive correspondence or resemblance between storytelling and programming as well as with respect to the end products characterizing such activities, support our claim that the domain of storytelling is a valid inspiration and certainly a reliable source of knowledge for software engineers. This correspondence, it can be argued, can contribute to add new perspectives, insights, and guidelines for reshaping standardized practices in computer science (such as programming). However, if – has we just demonstrated – there are important correspondences between storytelling and software development, why should not we go even deeper and investigate further perhaps, how general principles or purposes underlying the practice of storytelling can influence the practice of software engineering?

4.2. RQ2: What are the purposes of storytelling, especially as perceived by software engineers, during software development?

The application of some general principles underlying storytelling to the process of software development enabled the following considerations:

1. *capturing of tacit knowledge* via externalization, as for instance in pair programming, when the person controlling the keyboard has to explain the code to the other member of the pair;
2. *enhancing communication* among all people involved in the project – the stakeholders – by producing appropriate narratives and constantly collecting feedback; also by helping in supporting *team building* activities, and reinforcing social bonds via discussions and amendments to shared stories;
3. *better comprehension* of the developed system via shared metaphors, common project's vocabulary, and description of system's views;
4. *more creativity* in approaching problems and *more commitment* to the project via collaborative activities and specialities of storytelling techniques.

Such principles have significant practical implications for software engineers, especially with respect to gaining certain benefits (such as developing more compelling requirements, more detailed plans, more understandable code, higher-quality product, more successful project's presentations, and more coordinated, collaborative work).

Abstracting away from these considerations we can also interpret such results as evidence that the analogy with storytelling is not just established in theory, but can also bring about valuable insights for the field. In other words, it can equip software developers with a set of principles suitable for adoption and appropriate interpretation in their daily teaming practice, that is strongly dominated by social issues.

4.3. RQ3: What are the principles underlying storytelling that can help in refining current practices in software development?

We described a series of principles of storytelling that software practitioners could take into consideration in order to revise their daily practice. Ranging from elements to more coherent models, all the papers selected pointed out some beneficial effects of storytelling on software development. It turned out that while there was not a universal agreement on various proposals, each had its own standpoint and provided a unique set of suggestions. The combined analysis of all the proposals made suggested that:

1. Previous experience and predefined knowledge are necessary to achieve balance in software design.
2. System clarity is improved through the utilization of storytelling and architectural metaphors. Most of the principles given use phrases such as "*detailed setting*", "*clear guidance*", "*clear description*".
3. Textual descriptions and written reports with individual explanatory element aid programming projects.
4. When relaying requirements, inclusion of narratives with detailed plots and perspective helps to provide an intuitive understanding.

We believe that an interesting principle of storytelling that can effectively be used by software developers is: "*Show, do not tell*". This principle can be useful – for instance – in the following situations:

1. when a programmer writes a code, because the code is most useful when humans can understand it easily, and is a fact that programmers spend most of their time trying to understand codes [148]. when presenting a software product to some perspective users, this principle may help in contextualizing the user experience that the developers intend to convey [81]

5. Limitations, threats to validity and review assessment

In any scientific work, there is always room for improvement [149,150]. In this section we would like to discuss factors that might have influenced or undermined the objectivity, accuracy, and completeness of our findings. The analysis of potential shortcomings, affecting our study is of paramount importance not only to point out potential issues with our work but also, and perhaps more significantly, to better understand possible future research directions, as well as to let our readers think more critically about the topic we covered in this study.

5.1. Limitations

We start by looking at circumstances and factors limiting, hence potentially affecting the completeness of our investigation.

We believe the main limitation of this study in this respect lie in the restricted amount of data collected and processed. In particular, we can consider four factors as potentially, but not necessarily, problematic:

1. the fact that we searched a limited amount of databases (three). However, it can be argued that searches in these databases usually overlap with searches on other databases (e.g. Springer Link, Scopus), hence it is not necessary to formally check all available databases in order to find reliable data;

2. perhaps we adopted a stricter exclusion criterion than needed. In other words, a reader may question our choice to eliminate works on storytelling that did not have a textual requirement. However, the textual requirement in the study selected seems justified because we wanted to compare the practice of storytelling with the practice of programming or coding, which is textual in character.
3. perhaps one of the basic inclusion criteria adopted in our work (selecting only papers written in English) is constraining the cultural diversity on qualitative studies [151, 152]. We acknowledge this as a very good objection and as a very serious point of contention. However, we also note that much of the literature in the field is in English, so the requirement we adopted is not an unusual one for the literature.
4. a skeptical reader may well object that the topic we selected is too vague and borderline and that, as a consequence, there is not much work or interest on it as of now. Nevertheless, the scientific literature we found is worth of a careful analysis; we believe we managed to achieve this in our study, hence to provide an informed service for a broad readership in computer science.

5.2. Threats to validity

We now consider potential threats (or biases) that might call into question the overall credibility and validity of the conclusions presented in our study. According to Akl et al. (2019) [153], there are, at least, seven types of different biases potentially affecting any research: **(a)** publication bias; **(b)** time lag bias; **(c)** multiple (duplicate) publication bias; **(d)** location bias; **(e)** citation bias; **(f)** language bias; and **(g)** outcome reporting bias.

After a critical analysis we believe that only two of such biases may have potentially affected our work. These are:

1. the **publication bias** – We analyzed only papers that directly pointed out some role of storytelling principles in the practice of software development. We did not find studies demonstrating the absence of positive effects in the application of storytelling practices to software engineering.
2. the **language bias** – As noted above, we considered only papers written in English. Thus, in our searches we might have neglected relevant studies published in other languages, perhaps reporting less significant or negative results. We acknowledge this as a remote possibility though.

In our case, since most of us are employed by a private university, it is also worth mentioning one more type of bias — Industry Sponsorship bias [154]. Our research was not sponsored by any third-party company or organization and that, as a consequence, we feel that our review is free from potential external interference.

5.3. Review assessment

As a final step in the critical assessment of our findings, we answered four benchmark questions defining, in general, the overall quality of a systematic literature review [155]:

1. *Are the review's inclusion and exclusion criteria described and appropriate?* All criteria used for inclusion or exclusion were clearly mentioned in our protocol upfront (see methodology section). All the criteria, although certainly not perfect, are reasonable and coherent; hence we believe that they are appropriate for the study.
2. *Is the literature search likely to have covered all relevant studies?* As we already mentioned, there were certain limitations that prevented us from collecting all the possible data available out there. However, the process we establish to verify our protocol and the methodology we used to check our results was very through, sound, and comprehensive. Hence, we are confident that our work is scientifically sound. In other words, this condition is sufficiently met.
3. *Did the reviewers assess the quality/validity of the included studies?* We count this condition as partially met, because despite inclusion of credible and peer reviewed papers only, we did not (strictly speaking) set other quality criteria to assess the studies more precisely.
4. *Were the basic data/studies adequately described?* This condition is certainly met because we built a comprehensive reading log to put all the relevant information extracted from the papers we selected. Moreover, we also processed our data comprehensively in different ways. See Figs. 3, 4, 5 and Table 1, for instance.

6. Conclusion

We conducted a systematic literature review to investigate current research carried out at the overlapping of software development and storytelling. We had two major goals in this review: i. bringing software developers' attention to the existence of a new field of research that sees software development not just as an engineering process (a highly formalizable practice); but, rather as a creative activity or process. Using the PIECES framework [85], we clarified our research questions, which are as follow:

1. What are the similarities between the practice of software development and the practice of storytelling?
2. What are the specific goals of storytelling, especially as perceived by software engineers, during software development?
3. What are the principles underlying storytelling that can help in refining current practices in software development?

The focal point of our work was to find credible papers written in English that focused on practical applications of storytelling principles in software development. We searched four different repositories: ACM DL, Microsoft Academic, Google Scholar, and Research Gate. We used the PRISMA protocol/checklist [156] to filter out irrelevant documents and carry out a textual synthesis on the papers we included in our log. The relevance as well as the soundness of our results were then critically assessed, while their significance for the field was carefully discussed. Drawing conclusions and making recommendations, we see the following areas as providing promising insights for future research:

1. comprehensive mapping between stages of writing and stages of coding — for instance, how debugging and refactoring steps in programming are connected with the process of editing in writing;
2. common skills for writers/storytellers and software developers — thus focusing on which competences can mutually benefit experts from both domains;
3. new methodologies of teaching how to program and code — that is, appropriately incorporating in software engineering the best teaching practices found in storytelling;
4. roles of figures of speech in project's artifacts (such as requirements and documentation);
5. analysis of papers that consider not only written stories (textual narratives as we did here), but oral or digital stories as well and;
6. study of works and papers that experiment not only storytelling, but also other creative activities.

Further, we strongly believe that despite potential biases revealed in our review process, our results and findings are pretty reliable and indeed suggest that the adoption of certain principles of storytelling in software development can bring about numerous benefits to the discipline. In addition, we also think that this domain has the potential to (radically) change (probably for the better) how software practitioners work. Finally, despite our work is still preliminary, we sincerely hope that this review will broaden interest in this topic, hence provide new theoretical grounds for more detailed explorations into these extraordinarily rich and fascinating set of phenomena.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

We thank Innopolis University for generously funding this research. Paolo Ciancarini thanks CINI and ISTC-CNR for supporting his research.

Appendix

See Tables 2 and 3.

References

[1] Alexander Schmoelz, Enabling co-creativity through digital storytelling in education, Think. Skills Creat. 28 (2018) 1–13.
[2] Kevin Devaney, James Johnson, Storytelling as a key enabler for systems engineering, in: INCOSE International Symposium, Vol. 27, 2017, pp. 894–907, http://dx.doi.org/10.1002/j.2334-5837.2017.00401.x.
[3] Karin Thier, Storytelling in Organizations, Springer, 2018.
[4] Parissa Haghirian, Tina Chini, Storytelling: Transferring tacit corporate knowledge in different cultures, in: Proc. 2nd Annual European Academic Management (EURAM) Conference, Sweden, 2002.
[5] David West, Jenny Quillien, Patterns for story craft, in: Proc. 8th Latin American Conference on Pattern Languages of Programs, SugarLoafPLoP '10, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450302609, 2010, http://dx.doi.org/10.1145/2581507.2581508.
[6] Katie Elson Anderson, Storytelling, in: James Birx (Ed.), 21st Century Anthropology: A Reference Handbook, SAGE, 2010, Chap. 28.
[7] Austen Rainer, Storytelling in human–centric software engineering research, in: Proc. Int. Conf. on Evaluation and Assessment in Software Engineering, EASE, in: International Conference Proceeding Series, ACM, pp. 1–11.
[8] Raffaele Fabio Ciriello, Alexander Richter, Gerhard Schwabe, When prototyping meets storytelling: Practices and malpractices in innovating software firms, in: Proc. IEEE/ACM 39th Int. Conf. on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2017, pp. 163–172, http://dx.doi.org/10.1109/ICSE-SEIP.2017.24.
[9] Walter Parkes, Random access, remote control: the evolution of storytelling, Omni 16 (4) (1994) 48–54.
[10] John Feliks, A prehistory of hiking—Neanderthal storytelling, Pleistocene Coalit. News 3 (2) (2011) 1–2.
[11] Diana James, Signposted by song: Cultural routes of the Australian desert, Hist. Environ. 25 (3) (2013) 30–42.
[12] Enzo Caminotti, Jeremy Gray, The effectiveness of storytelling on adult learning, J. Workplace Learn. (2012).
[13] Donald Miller, Building a Storybrand: Clarify Your Message so Customers Will Listen, HarperCollins Leadership, 2017.
[14] Robert McKee, Thomas Gerace, Storynomics: Story-Driven Marketing in the Post-Advertising World, Hachette UK, 2018.
[15] Jerome Bruner, History of science & technology acts of meaning, Bull. Sci. Technol. Soc. 13 (1) (1993) 57, http://dx.doi.org/10.1177/027046769301300193.
[16] William Hanks, Text and textuality, Ann. Rev. Anthropol. (ISSN: 00846570) 18 (1989) 95–127, URL: http://www.jstor.org/stable/2155887.
[17] Carolyn S. Phillips, Deborah L. Volker, Kristin L. Davidson, Heather Becker, Storytelling through music: A multidimensional expressive arts intervention to improve emotional well-being of oncology nurses, JCO Oncol. Pract. 16 (4) (2020) e405–e414.
[18] Karin Eli, Rosie Kay, Choreographing lived experience: dance, feelings and the storytelling body, Med. Humanit. 41 (1) (2015) 63–68.
[19] Jason Lankow, Josh Ritchie, Ross Crooks, Infographics: The Power of Visual Storytelling, Wiley, 2012.
[20] Ruth Sylvester, Wendy-lou Greenidge, Digital storytelling: Extending the potential for struggling writers, Read. Teacher 63 (4) (2009) 284–295.
[21] Kath Dooley, Storytelling with virtual reality in 360-degrees: a new screen grammar, Stud. Aust. Cine. 11 (3) (2017) 161–171.
[22] Matthew Freeman, Historicising Transmedia Storytelling: Early Twentieth-Century Transmedia Story Worlds, Routledge, 2016.
[23] Carolyn Handler Miller, Digital Storytelling: A Creator's Guide to Interactive Entertainment, Routledge, 2014.
[24] Alejandra Recio, Task-based instruction and storytelling with young learners: Analysis of its advantages, Greta Mag. 13 (1&2) (2005) 61–70.
[25] Michael D. Jones, Holly Peterson, Narrative persuasion and storytelling as climate communication strategies, in: Oxford Research Encyclopedia of Climate Science, 2017.
[26] Phil Hopkins, Mass Moralizing: Marketing and Moral Storytelling, Lexington Books, 2015.
[27] Linda Flower, John R. Hayes, A cognitive process theory of writing, Coll. Compos. Commun. (ISSN: 0010096X) 32 (4) (1981) 365–387.
[28] D. Gordon Rohman, Pre-writing the stage of discovery in the writing process, Coll. Compos. Commun. (ISSN: 0010096X) 16 (2) (1965) 106–112.
[29] Henk Huizenga, Taal & Didactiek: Stellen, Noordhoff Uitgevers BV, 2004.
[30] Edward F. Pace-Schott, Dreaming as a story-telling instinct, Front. Psychol. 4 (2013) 159.
[31] Nancy Lamb, The Art and Craft of Storytelling: A Comprehensive Guide to Classic Writing Techniques, Writer's Digest Books, ISBN: 1582975590, 2008.
[32] Linde Zingaro, Speaking Out: Storytelling for Social Change, Routledge, 2017.
[33] Klaus Fog, et al., Storytelling: Branding in Practice, Springer, 2010.
[34] Klaus Fog, Christian Budtz, Philip Munch, Stephen Blanchette, Storytelling - Branding in Practice, second ed., Springer, 2010.
[35] Will Storr, The Science of Storytelling, Abrams Press, 2020.
[36] Gene Kim, Kevin Behr, George Spafford, The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win, IT Revolution, 2013.
[37] Abram Hindle, Earl T. Barr, Mark Gabel, Zhendong Su, Premkumar Devanbu, On the naturalness of software, Commun. ACM 59 (5) (2016) 122–131.
[38] IEEE, Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, 1990, pp. 1–84, http://dx.doi.org/10.1109/IEEESTD.1990.101064.
[39] Brian W. Kernighan, Phillip J. Plauger, Software tools, ACM SIGSOFT Softw. Eng. Notes 1 (1) (1976) 15–20.
[40] Pedro Isaias, Tomayess Issa, Introduction to information systems models and methodologies, in: High Level Models and Methodologies for Information Systems, Springer New York, New York, NY, ISBN: 978-1-4614-9254-2, 2015, pp. 1–19, http://dx.doi.org/10.1007/978-1-4614-9254-2_1.
[41] Linda Sherrell, Waterfall model, in: Anne L.C. Runehov, Lluis Oviedo (Eds.), Encyclopedia of Sciences and Religions, Springer Netherlands, Dordrecht, ISBN: 978-1-4020-8265-8, 2013, pp. 2343–2344, http://dx.doi.org/10.1007/978-1-4020-8265-8_200285.
[42] Philippe Kruchten, The Rational Unified Process–An Introduction, third ed., Addison-Wesley, 2003.
[43] Mark Mahoney, Collaborative software development through reflection and storytelling, in: Proc. ACM Conference on Computer Supported Cooperative Work and Social Computing, 2017, pp. 13–16.
[44] Neil Maiden, Creativity in software engineering: a new research agenda? in: Proc. 18th Int. Conf. on Program Comprehension, IEEE, 2010, p. xiv.
[45] Mingyang Gu, Xin Tong, Towards hypotheses on creativity in software development, in: Proc. Int. Conference on Product Focused Software Process Improvement, Springer, 2004, pp. 47–61.
[46] Oliver Sacks, Tourette's syndrome and creativity, BMJ: Br. Med. J. 305 (6868) (1992) 1515.
[47] Mihaly Csikszentmihalyi, The domain of creativity, in: M. Runco, R. Albert (Eds.), Theories of Creativity, in: Sage focus editions, vol. 115, Sage Publications, Inc, 1990, pp. 190–212.
[48] Mihaly Csikszentmihalyi, Society, culture, and person: A systems view of creativity, in: The Systems Model of Creativity, Springer, 2014, pp. 47–61.
[49] Mihaly Csikszentmihalyi, The Systems Model of Creativity: The Collected Works of Mihaly Csikszentmihalyi, Springer, 2015.

[50] Ronald Leach, Caprice Ayers, The psychology of invention in computer science, in: Proc. 17th Annual Workshop of the Psychology of Programming Interest Group, PPIG, University of Sussex, Brighton, 2005.

[51] Robert Glass, Tom DeMarco, Software Creativity 2.0, in: Online Access: EBSCO Computers & Applied Sciences Complete, Developer.* Books, ISBN: 9780977213313, 2006, URL: https://books.google.ru/books?id=DozsD0zxb5wC.

[52] Anita Lie, Paired storytelling: An integrated approach for EFL students, J. Read. 36 (8) (1993) 656–658.

[53] Kent Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley professional, 2000.

[54] Roger S. Pressman, Software Engineering: A Practitioner's Approach, European, McGraw-Hill, 1994.

[55] Ilya Khomyakov, Sergey Masyagin, Giancarlo Succi, Experience of mixed learning strategies in teaching lean software development to third year undergraduate students, in: International Workshop on Frontiers in Software Engineering Education, Springer, 2020, pp. 42–59.

[56] Felienne Hermans, Marlies Aldewereld, Programming is Writing is Programming, Programming '17, ACM, Brussels, Belgium, ISBN: 9781450348362, 2017.

[57] Andreas Munk-Madsen, Peter Andersen, Storytelling in projects: transforming project plans into stories, Hentet Mars 12 (2006).

[58] Kevin Devaney, James Johnson, Storytelling as a key enabler for systems engineering, in: INCOSE International Symposium, Vol. 27, 2017, pp. 894–907, http://dx.doi.org/10.1002/j.2334-5837.2017.00401.x.

[59] Naoufel Boulila, Anne Hoffmann, Andrea Herrmann, Using storytelling to record requirements: Elements for an effective requirements elicitation approach, in: Proc. 4th Int. Workshop on Multimedia and Enjoyable Requirements Engineering, MERE, 2011, pp. 9–16, http://dx.doi.org/10.1109/MERE.2011.6043945.

[60] Rubia Fatima, Affan Yasin Chouhan, Lin Liu, Jianmin Wang, Wasif Afzal, Atif Yasin, Improving software requirements reasoning by novices: A story-based approach, IET Softw. 13 (6) (2019) 564–574, http://dx.doi.org/10.1049/iet-sen.2018.5379.

[61] Mary Elizabeth Jones, Melanie Kisthardt, Marie A. Cooper, Interdisciplinary teaching: Introductory programming via creative writing, in: Proc. 42nd ACM Technical Symposium on Computer Science Education, SIGCSE '11, ACM, New York, NY, USA, ISBN: 9781450305006, 2011, pp. 523–528, http://dx.doi.org/10.1145/1953163.1953313.

[62] Nik Boyd, Software metaphors, 2003, http://www.educery.com/papers/rhetoric/metaphors.

[63] Evgeny Pyshkin, Metaphor models in software education: An empirical study, in: Proc. Int. Conf. on Sw Engineering Advances, ICSEA, 2019, pp. 30–35.

[64] Clarisse Herrenschmidt, Les Trois écritures Langue, Nombre, Code, Gallimard, 2007.

[65] Sally Yeates Sedelow, The computer in the humanities and fine arts, ACM Comput. Surv. 2 (2) (1970) 89–110.

[66] Donald Knuth, Literate programming, Comput. J. 27 (2) (1984) 97–111.

[67] Donald Knuth, The Art of Computer Programming, Vol. 3, Pearson Education, 1997.

[68] Jonathan Wallace, Is Software Art or Engineering? 1999, URL: https://www.spectacle.org/1199/software.html.

[69] Donald Knuth, The art of programming, ITNow 53 (4) (2011).

[70] Florian Cramer, Ulrike Gabriel, Software art, in: Alan Sondheim (Ed.), American Book Review, Issue "Codeworks", 2001.

[71] Paul Fishwick, et al., Aesthetic computing "manifesto", Leonardo 36 (4) (2003) 255–256.

[72] Gregory W. Bond, Software as art, Commun. ACM 48 (8) (2005) 118–124.

[73] Paul Fishwick (Ed.), Aesthetic Computing, MIT Press, 2008.

[74] Anna Trifonova, Letizia Jaccheri, Kristin Bergaust, Software engineering issues in interactive installation art, Int. J. Arts Technol. 1 (1) (2008) 43–65.

[75] Irina Erofeeva, Vladimir Ivanov, Sergey Masyagin, Giancarlo Succi, Learning agility from dancers - experience and lesson learnt, in: Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment - Second International Workshop, DEVOPS 2019, ChâTeau de Villebrumier, France, May 6-8, 2019, Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 12055, Springer, 2019, pp. 112–120.

[76] Sergey Masyagin, Milana Nurgalieva, Giancarlo Succi, Kent beck or pablo picasso? Speculations of the relationships between artists in software and painting, in: Software Technology: Methods and Tools - 51st International Conference, TOOLS 2019, Innopolis, Russia, October 15-17, 2019, Proceedings, in: Lecture Notes in Computer Science, vol. 11771, Springer, 2019, pp. 3–9.

[77] Paolo Ciancarini, Sergey Masyagin, Giancarlo Succi, Software design as story telling: reflecting on the work of Italo Calvino, in: Proc. ACM SIGPLAN Int Symp on New Ideas New Paradigms, and Reflections on Programming and Software - SPLASH, ACM, 2020, pp. 195–208.

[78] Arlene Fink, Conducting Research Literature Reviews: From the Internet to Paper, Sage publications, 2019.

[79] Gerard P. Hodgkinson, J. Kevin Ford, Narrative, meta-analytic, and systematic reviews: What are the differences and why do they matter? J. Organ. Behav. 35 (S1) (2014) S1–S5.

[80] Pascal Schneider, An agile methodology for greater communications excellence in the age of digitalization: Communications "coding" loop, in: Proc. International Professional Communication Conference (ProComm), IEEE, 2019, pp. 164–170.

[81] Whitney Quesenbery, Kevin Brooks, Storytelling for User Experience: Crafting Stories for Better Design, Rosenfeld Media, 2010.

[82] Wayne Lutters, Carolyn Seaman, Revealing actual documentation usage in software maintenance through war stories, Inf. Softw. Technol. 49 (6) (2007) 576–587.

[83] M.R.K. Krishna Rao, Storytelling and puzzles in a software engineering course, ACM SIGCSE Bull. 38 (1) (2006) 418–422.

[84] Barbara Ann Kitchenham, Stuart Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Vol. 2, Tech. rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[85] Margaret Foster, Systematic reviews service: Introduction to systematic reviews, 2018, URL: https://tamu.libguides.com/c.php?g=574702&p=4298724.html.

[86] Adrian Sayers, Tips and tricks in performing a systematic review, Br. J. Gen. Pract. 58 (547) (2008) 136.

[87] Connie Schardt, Martha B. Adams, Thomas Owens, Sheri Keitz, Paul Fontelo, Utilization of the PICO framework to improve searching PubMed for clinical questions, BMC Med. Inform. Decis. Mak. 7 (1) (2007) 1–6.

[88] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, Mohamed Khalil, Lessons from applying the systematic literature review process within the software engineering domain, J. Syst. Softw. 80 (4) (2007) 571–583.

[89] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, Stephen Linkman, Systematic literature reviews in software engineering–a systematic literature review, Inf. Softw. Technol. 51 (1) (2009) 7–15.

[90] Mirko Farina, Maxim Kostin, Giancarlo Succi, Interest identification from browser tab titles: A systematic literature review, Comput. Human Behav. Rep. (ISSN: 2451-9588) (2022) 100187, http://dx.doi.org/10.1016/j.chbr.2022.100187, URL: https://www.sciencedirect.com/science/article/pii/S2451958822000215.

[91] Mirko Farina, Arina Fedorovskaya, Egor Polivtsev, Giancarlo Succi, Software engineering and filmmaking: a literature review, Front. Comput. Sci. (2022) 53.

[92] Mirko Farina, Anna Gorb, Artem Kruglov, Giancarlo Succi, Technologies for GQM-based metrics recommender systems: A systematic literature review, IEEE Access 10 (2022) 23098–23111, http://dx.doi.org/10.1109/ACCESS.2022.3152397.

[93] David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G. Altman, Prisma Group, et al., Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement, PLoS Med. 6 (7) (2009) e1000097.

[94] Gianluigi Caldiera, Victor Basili, Dieter Rombach, The goal question metric approach, in: John Marciniak (Ed.), Encyclopedia of Software Engineering, Wiley, 1994, pp. 528–532.

[95] Daniel Méndez Fernández, Jan-Hendrik Passoth, Empirical software engineering: from discipline to interdiscipline, J. Syst. Softw. 148 (2019) 170–179.

[96] Kai Petersen, Sairam Vakkalanka, Ludwik Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Inf. Softw. Technol. 64 (2015) 1–18.

[97] Cecilia Patino, Juliana Ferreira, Inclusion and exclusion criteria in research studies: definitions and why they matter, J. Bras. Pneumol. 44 (2018) 84, http://dx.doi.org/10.1590/s1806-37562018000000088.

[98] Hasse Clausen, Designing computer systems from a human perspective: the use of narratives, Scand. J. Inf. Syst. 6 (2) (1994) 1.

[99] Tore Dybå, Torgeir Dingsøyr, Geir Hanssen, Applying systematic reviews to diverse study types: An experience report, in: Proc. 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, ISBN: 978-0-7695-2886-1, 2007, pp. 225–234, http://dx.doi.org/10.1109/ESEM.2007.59.

[100] Patricia Lucas, Janis Baird, Lisa Arai, Conan Law, Helen Roberts, Worked examples of alternative methods for the synthesis of qualitative and quantiative research in systematic reviews, BMC Med. Res. Methodol. 7 (2007) 4, http://dx.doi.org/10.1186/1471-2288-7-4.

[101] Ziva Hassenfeld, Marina Bers, Debugging the writing process: Lessons from a comparison of students' coding and writing practices, Read. Teacher 73 (2020) 735–746.

[102] Caitlin Kelleher, Randy Pausch, Sara Kiesler, Storytelling alice motivates middle school girls to learn computer programming, in: Proc. ACM SIGCHI Conference on Human FFactors in Computing Systems, 2007, pp. 1455–1464.

[103] Raphael Perret, Marcos R.S. Borges, Flávia Maria Santoro, Applying group storytelling in knowledge management, in: Gert-Jan de Vreede, Luis A. Guerrero, Gabriela Marín Raventós (Eds.), Groupware: Design, Implementation, and Use, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-540-30112-7, 2004, pp. 34–41.
[104] Murat Yilmaz, Berke Atasoy, Rory V. O'Connor, Jean-Bernard Martens, Paul Clarke, Software developer's journey, in: Christian Kreiner, Rory V. O'Connor, Alexander Poth, Richard Messnarz (Eds.), Systems, Software and Services Process Improvement, Springer International Publishing, Cham, ISBN: 978-3-319-44817-6, 2016, pp. 203–211.
[105] Line Dubé, Daniel Robey, Software stories: three cultural perspectives on the organizational practices of software development, Account. Manag. Inf. Technol. 9 (4) (1999) 223–259.
[106] David Snowden, Story telling: an old skill in a new context, Bus. Inf. Rev. 16 (1) (1999) 30–37.
[107] Peter Lloyd, Storytelling and the development of discourse in the engineering design process, Des. Stud. 21 (4) (2000) 357–373.
[108] Stefana Broadbent, Francesco Cara, A narrative approach to user requirements for web design, Interactions 7 (6) (2000) 31–35.
[109] Dan Gruen, Thyra Rauch, Sarah Redpath, Stefan Ruettinger, The use of stories in user experience design, Int. J. Human-Comput. Interact. 14 (3–4) (2002) 503–534.
[110] Mike Cohn, User Stories Applied: For Agile Software Development, Addison-Wesley Professional, 2004.
[111] Tammy VanDeGrift, Coupling pair programming and writing: learning about students' perceptions and processes, in: Proc. 35th SIGCSE Technical Symposium on Computer Science Education, ACM, Norfolk, Virginia, USA, 2004, pp. 2–6.
[112] Brian Bussell, Stephen Taylor, Software development as a collaborative writing project, in: International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer, 2006, pp. 21–31.
[113] Adriana Cristina de Oliveira, Renata Mendes de Araujo, Marcos RS Borges, Telling stories about system use: Capturing collective tacit knowledge for system maintenance, in: Proc. Int. Conf. on Sw Eng. and Knowledge Eng., SEKE, 2007, pp. 337–342.
[114] James Siddle, "Choose your own architecture"-interactive pattern storytelling, in: Transactions on Pattern Languages of Programming II, Springer, 2011, pp. 16–33.
[115] Sean Hammond, Tim J. Smith, Helen Pain, Children's story authoring with propp's morphology: An exploratory study, in: 5th International Conference on Narrative and Interactive Learning Environments Edinburgh, Scotland 6th–8th August 2008, 2008.
[116] Roger McDermott, Gordon Eccleston, Garry Brindley, More than a good story—can you really teach programming through storytelling? Innov. Teach. Learn. Inf. Comput. Sci. 7 (1) (2008) 34–43.
[117] Stephen C. Hayne, Using storytelling to enhance information systems knowledge transfer, in: ISOneWorld Conference, 2009.
[118] Erik Wende, Parissa Haghirian, Storytelling as a tool for knowledge transfer in the IT industry, in: Proc. 17th European Conference on Information Systems, ECIS, Verona, Italy, 2009.
[119] Viviane Laporti, Marcos R.S. Borges, Vanessa Braganholo, Athena: A collaborative approach to requirements elicitation, Comput. Ind. 60 (6) (2009) 367–380.
[120] Sabine Madsen, Lene Nielsen, Exploring persona-scenarios using storytelling to create design ideas, in: Proc. IFIP Working Conference on Human Work Interaction Design, Springer, 2009, pp. 57–66.
[121] Debra Richardson, Ban Al-Ani, Hadar Ziv, Requirements engineering at the margins: avoiding technological hubris through alternative approaches, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, 2010, pp. 303–308.
[122] Quinn Burke, Yasmin B. Kafai, Programming & storytelling: opportunities for learning about coding & composition, in: Proc. 9th Int. Conf. on Interaction Design and Children, 2010, pp. 348–351.
[123] Hamizah Mohamad Hariri, Abu Bakar Marini, Abdullah Mohd Zin, Story telling approach for integrating software blocks, in: Proc. International Conference on Electrical Engineering and Informatics, IEEE, 2011, pp. 1–5.
[124] Roman Knöll, Vaidas Gasiunas, Mira Mezini, Naturalistic types, in: Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, 2011, pp. 33–48.
[125] Michael Keeling, Michail Velichansky, Making metaphors that matter, in: Proc. Agile Conference, IEEE, 2011, pp. 256–262.
[126] Madeleine Kusoffsky, Leveraging Storytelling in Visual Analytics by Redesigning the User Interface (Thesis), The Institute of Technology, Linkoping University, 2013.
[127] Quinn Burke, The markings of a new pencil: Introducing programming-as-writing in the middle school classroom, J. Media Lit. Educ. 4 (2) (2012) 121–135.
[128] Philip Van Allen, Joshua McVeigh-Schultz, Brooklyn Brown, Hye Mi Kim, Daniel Lara, AniThings: animism and heterogeneous multiplicity, in: CHI'13 Extended Abstracts on Human Factors in Computing Systems, 2013, pp. 2247–2256.
[129] Arjen Uittenbogaard, Storytelling for software professionals, IEEE Softw. 30 (3) (2013) 9–12.
[130] Erik Wende, Gregory King, Gerhard Schwabe, Exploring storytelling as a knowledge transfer technique in offshore outsourcing, in: Proc. Int. Conf. on Information Systems, 2014.
[131] Azad Madni, Marcus Nance, Michael Richey, William Hubbard, Leroy Hanneman, Toward an experiential design language: Augmenting model-based systems engineering with technical storytelling in virtual worlds, Procedia Comput. Sci. (ISSN: 1877-0509) 28 (2014) 848–856, http://dx.doi.org/10.1016/j.procs.2014.03.101, Proc. Conference on Systems Engineering Research. URL: https://www.sciencedirect.com/science/article/pii/S1877050914001641.
[132] Angelo Gaeta, Matteo Gaeta, Giuseppe Guarino, Sergio Miranda, A smart methodology to improve the story-building process, J. E-Learn. Knowl. Soc. 11 (1) (2015).
[133] Manuel Brhel, Hendrik Meth, Alexander Maedche, Karl Werder, Exploring principles of user-centered agile software development: A literature review, Inf. Softw. Technol. 61 (2015) 163–181.
[134] Kimberly A. Gausepohl, Woodrow W. Winchester, Tonya L. Smith-Jackson, Brian M. Kleiner, James D. Arthur, A conceptual model for the role of storytelling in design: leveraging narrative inquiry in user-centered design (UCD), Health Technol. 6 (2) (2016) 125–136.
[135] Nguyen Thien Khanh, Jirapun Daengdej, Habibi Husain Arifin, Human stories: A new written technique in agile software requirements, in: Proc. 6th International Conference on Software and Computer Applications, ICSCA '17, ACM, Bangkok, Thailand, 2017, pp. 15–22, http://dx.doi.org/10.1145/3056662.3056680.
[136] Edhy Rustan, Learning creative writing model based on neurolinguistic programming, Int. J. Lang. Educ. Cult. Rev. 3 (2) (2017) 13–29.
[137] Zahra Shakeri Hossein Abad, Alex Shymka, Jenny Le, Noor Hammad, Guenther Ruhe, A visual narrative path from switching to resuming a requirements engineering task, in: Proc. IEEE 25th Int. Conf. on Requirements Engineering, RE, IEEE, 2017, pp. 442–447.
[138] Chao Tong, Richard Roberts, Rita Borgo, Sean Walton, Robert S. Laramee, Kodzo Wegba, Aidong Lu, Yun Wang, Huamin Qu, Qiong Luo, et al., Storytelling and visualization: An extended survey, Information 9 (3) (2018) 65.
[139] Rodi Jolak, Maxime Savary-Leblanc, Manuela Dalibor, Andreas Wortmann, Regina Hebig, Juraj Vincur, Ivan Polasek, Xavier Le Pallec, Sébastien Gérard, Michel R.V. Chaudron, Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication, Empir. Softw. Eng. 25 (6) (2020) 4427–4471.
[140] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, David Moher, Updating guidance for reporting systematic reviews: development of the PRISMA 2020 statement, J. Clin. Epidemiol. 134 (2021) 103–112.
[141] Michael Keeling, Michail Velichansky, Making metaphors that matter, in: Proc. Agile Conference, IEEE, 2011, pp. 256–262.
[142] Paolo Ciancarini, Mirko Farina, Sergey Masyagin, Giancarlo Succi, Sofiia Yermolaieva, Nadezhda Zagvozkina, Root causes of interaction issues in agile software development teams–status and perspectives, Adv. Intell. Syst. Comput. 2 (2021) 1017–1036, http://dx.doi.org/10.1007/978-3-030-73103-8_74.
[143] Paolo Ciancarini, Mirko Farina, Sergey Masyagin, Giancarlo Succi, Sofiia Yermolaieva, Nadezhda Zagvozkina, Non verbal communication in software engineering–an empirical study, IEEE Access 9 (2021) 71942–71953, http://dx.doi.org/10.1109/ACCESS.2021.3075983.
[144] Dan Gruen, Thyra Rauch, Sarah Redpath, Stefan Ruettinger, The use of stories in user experience design, Int. J. Human–Comput. Interact. 14 (3–4) (2002) 503–534, http://dx.doi.org/10.1080/10447318.2002.9669132.
[145] Hasse Clausen, Designing computer systems from a human perspective: The use of narratives, Scand. J. Inf. Syst. (ISSN: 0905-0167) 6 (2) (1994) 43–58.
[146] H. Porter Abbott, The Cambridge Introduction to Narrative, Cambridge University Press, 2020.
[147] Erich Gamma, Design patterns—ten years later, in: Manfred Broy, Ernst Denert (Eds.), Software Pioneers, Springer, 2002, pp. 688–700.
[148] Simone Scalabrino, Gabriele Bavota, Christopher Vendome, Mario Linares-Vasquez, Denys Poshyvanyk, Rocco Oliveto, Automatically assessing code understandability, IEEE Trans. Softw. Eng. 47 (3) (2019) 595–613.
[149] Alexander Bird, What is scientific progress? Noûs 41 (1) (2007) 64–89.
[150] Daniel N. Robinson, Paradigms andthe myth of framework' how science progresses, Theory Psychol. 10 (1) (2000) 39–47.
[151] Joseph Henrich, Steven J. Heine, Ara Norenzayan, The weirdest people in the world? Behav. Brain Sci. 33 (2–3) (2010) 61–83.

[152] Joseph Henrich, The Weirdest People in the World: How the West Became Psychologically Peculiar and Particularly Prosperous, Farrar, Straus and Giroux, 2020.

[153] Elie Akl, Douglas Altman, Patricia Aluko, Lisa Askie, Dorcas Beaton, Jesse Berlin, Bhaumik Bhaumik, Clifton Bingham, Maarten Boers, Andrew Booth, Isabelle Boutron, Sue Brennan, Matthias Briel, Simon Briscoe, Jason Busse, Deborah Caldwell, Margaret Cargo, Alonso Carrasco-Labra, Anna Chaimani, Camilla Young, Cochrane Handbook for Systematic Reviews of Interventions, John Wiley & Sons, ISBN: 9781119536604, 2019, http://dx.doi.org/10.1002/9781119536604.

[154] B. Holman, Bero L., Mintzes B., Industry sponsorship bias. Catalogue of bias 2019, 2019, URL: https://catalogofbias.org/biases/industry-sponsorship-bias.html.

[155] Barbara Kitchenham, Procedures for Performing Systematic Reviews, Tech. rep. 33, Keele Univ, UK, 2004.

[156] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, Roger Chou, Julie Glanville, Jeremy M. Grimshaw, Asbjörn Hróbjartsson, Manoj M. Lalu, Tianjing Li, Elizabeth W. Loder, Evan Mayo-Wilson, Steve McDonald, Luke A. McGuinness, Lesley A. Stewart, James Thomas, Andrea C. Tricco, Vivian A. Welch, Penny Whiting, David Moher, The PRISMA 2020 statement: An updated guideline for reporting systematic reviews, Int. J. Surg. (ISSN: 1743-9191) 88 (2021) 105906, http://dx.doi.org/10.1016/j.ijsu.2021.105906, URL: https://www.sciencedirect.com/science/article/pii/S1743919121000406.