frontiers | Frontiers in Computer Science

SYSTEMATIC REVIEW
published: 29 April 2022
doi: 10.3389/fcomp.2022.884533

Software Engineering and Filmmaking: A Literature Review

Mirko Farina[1]*, *Arina Fedorovskaya*[2]*, *Egor Polivtsev*[2]* and *Giancarlo Succi*[2]*

[1] Faculty of Humanities and Social Sciences, Innopolis University, Innopolis, Russia, [2] Faculty of Computer Science and Engineering, Innopolis University, Innopolis, Russia

Software development is a complex process that requires skills in mathematics and physics. Moreover, it usually includes collaboration with other people. To get a precise understanding of the way such a process is organized, we need to understand its essence. Technical knowledge is crucially important for any developer; however, another important characteristic of any software engineer is creativity. In this article, we look at one particular artistic practice [filmmaking] that involves both these latter characteristics to determine whether insights from such a practice can be applied in the IT industry and vice versa.

Keywords: software development, movie production, lean, agile, software management practices

OPEN ACCESS

Edited by:
Jun Shen,
University of Wollongong, Australia

Reviewed by:
Sandra Sanchez-Gordon,
Escuela Politécnica Nacional, Ecuador
Luigi Benedicenti,
University of New Brunswick
Fredericton, Canada

***Correspondence:**
Mirko Farina
farinamirko@gmail.com
Arina Fedorovskaya
a.fedorovskaya@innopolis.ru
Egor Polivtsev
e.polivtsev@innopolis.ru
Giancarlo Succi
giancarlo.succi@gmail.com

Specialty section:
This article was submitted to Human-Media Interaction, a section of the journal Frontiers in Computer Science

Received: *26 February 2022*
Accepted: *05 April 2022*
Published: *29 April 2022*

Citation:
Farina M, Fedorovskaya A, Polivtsev E and Succi G (2022) Software Engineering and Filmmaking: A Literature Review. Front. Comput. Sci. 4:884533. doi: 10.3389/fcomp.2022.884533

1. INTRODUCTION

The relationship between software and art is one that has caused many interesting debates among computer scientists since, at least, the mid 70's (Sedelow, 1970), when a group of researchers, led by Donald Knuth, first argued that programming is itself an art (Knuth, 1984). Throughout his career, Knuth further defended this intuition (Knuth, 1997). More specifically, in a famous lecture titled "God and Computers," held at MIT in 1999, Knuth compared the beauty found in programming with that typically observed in literature or music and argued for the presence of, what he called, a sense of software aesthetics among computer scientist (Ahmed et al., 2009).

Knuth's approach to characterize the relationship between software and art became very influential. For example, Cramer and Gabriel (2001) extended Knuth's original intuition and focused on studying the beauty underlying the production of a code. Bond (2005) claimed that software can be considered as an artistic medium and Graham (2004) successfully discussed the relations between artistic works/behaviors and hacking, programming languages, and many other technological issues.

Other researchers too claimed that software engineering should be considered as art (Wallace, 1999). Jonathan Wallace, for example, argued that despite the fact that software development now occurs—for the most part at least—in teams; one still has the opportunity to show some artistic creativity in the solutions of certain problems. Wallace also noted that certain branches of software thrive in an artistic environment. For instance, he noticed how spreadsheets, databases, and browsers, were originally developed by an "artist" (or a group of artists), who created their initial prototype and then researched how to turn them into a product. In this sense then, artwork can be later transformed into an engineering product.

More recently, Fishwick et al. (2003) explored the way art and aesthetics can coalesce to increase innovation and creativity among software developers, while Fishwick (2008) further specified the enhancing role they can both play in different areas of computer science. On a similar vein, Trifonova et al. (2008) studied the extent to which software development can be enriched by embracing a multidisciplinary approach that converges at the intersection between art and computing.

In general, we can say that there is a relatively small but growing literature on these issues (Knuth, 2011) and that the field is ripe for further explorations.

Building up on this debate, our study is concerned with the analysis of software development techniques, their application to filmmaking, and the role of teamwork in movie production (Donelan, 2007). Our aim is thus to systematically compare the practice of filmmaking with the work carried out by software engineers. Before we go on to specify our motivation, goals, and contribution to the literature in more details, it is nevertheless worth noting that, as the topic of this investigation is relatively new, the academic literature available on it is quite limited.

The first step in producing an analysis capable of describing the relations between filmmaking and software development involves, we believe, the acknowledgment that modern film production, like software development, constitutively depends on successful teamwork (Kanaan, 2016) and on many different collaborative practises (Ohanian and Phillips, 2013). However, the analogy does not end here. Naturally, some techniques or management models used in software development are also profitably used by filmmakers (Agrawal, 2016).

Two of such techniques or models are of particular interest for us in this work. These are: agile (Beck et al., 2001), and lean (Netland and Powell, 2016). Agile software development can be described as non-traditional set of methodologies based on collaborative efforts, which aim to maximize production and efficiency by implementing: (i) quick responses to changes, (ii) cross-team interactions, and (iii) work-processes simplicity (Dingsøyr et al., 2010). The well-known Agile Manifesto (Beck et al., 2001), in the early 2000s, described the basic values and principles underlying Agile Software Development (Beck et al., 2001).

The idea of lean management firstly arose from the manufacturing environment (Shah and Ward, 2003), subsequently becoming a philosophy for both product and service industries (Janes and Succi, 2014). The Lean model can be described by three basic principles (Feld, 2000). These are:

- value: methods that support maximization of delivered value;
- knowledge: methods that focus on the creation of a shared understanding of the know-how, know-where, know-who, know-what, know-when, and know-why within the company;
- improvement: methods that instil a culture of constant improvement among and between team members.

Recently, Moreira (2020) successfully used lean concepts and principles (Poppendieck and Cusumano, 2012); Ebert et al. (2012) to bring assertiveness and increase team's performance in filmmaking.

The motivation for this study is to improve and enrich existing software development methodologies in light of contemporary practices characterizing the process of filmmaking. To do so, we distinguished three major steps in our work:

1. Definition of common features between software development and filmmaking.
2. Detailed analysis of resemblance between software development and movie production in respect to certain techniques.
3. Comparison, focused on usage of collaborative techniques (based on cooperation), between software development and filmmaking.

The main contribution of this work is therefore to:

- analyse film production techniques and their possible adaptation in software development.
- describe and comprehend the relevance of collaborative filmmaking for software engineering.
- provide parallels between the artistic work of filmmakers and the artistic work underlying the process of software development.
- create, on these grounds, potentially new techniques for software development

The novelty of this review consists in the fact that it aims to provide an original characterization of the relationship between film production and software development. Crucially, such a relationship has not been sufficiently studied by researchers so far.

This review is then organized as follows. Section 2 reviews the existing literature on filmmaking and software development, describing the research questions underlying our work, as well as the protocol, and the search strategy adopted in our study. Section 3 presents our findings and contextualizes them, focusing on possible parallels between film production and software development. Section 4 offers a brief synoptic summary of our results for the reader, while Section 5 describes shortcomings, limitations, and potential issues threatening the validity of our findings. Finally, section 6 explains the relevance of our findings for the field and outlines possible future research directions.

2. METHODOLOGY

We conducted a literature review in accordance with the suggestions outlined by Kitchenham (2004), Brereton et al. (2007), Kitchenham et al. (2009), and Siddaway (2014), and in conformity with the "Preferred Reporting Items for Systematic reviews and Meta-Analyses" (PRISMA) checklist[1] (Brereton et al., 2007; Moher et al., 2009), detailed in **Table 7**. **Table 1** describes our study protocol.

2.1. Research Questions

Our research questions concern the relations between filmmaking and software engineering. More specifically, our goal is to look for analogies between the processes underlying the production of basic software and those characterizing the creation and production of a movie. In particular, we focus on finding analogies between both areas to proof that filmmaking

[1] http://www.prisma-statement.org

TABLE 1 | Study protocol.

Phase	Stages
Review plan	1. Statement of research questions 2. Creation of review plan 3. Application of review plan
Review process	1. Identification of relevant studies 2. Selection of primary studies 3. Quality check on selected studies 4. Extraction of required information 5. Data analysis
Reporting	1. Creation of a report based on the data analyzed

and software development indeed share a common nature. Our article thus attempt to answer the following research questions:

- **RQ1**: How do processes characterizing filmmaking resemble practices and processes implemented in the software industry?

To answer this question we reviewed a series of filmmaking practices and programming techniques and then critically assessed the possibility to adapt such practices and techniques in the respective fields.

- **RQ2**: Which practices used in filmmaking, and not yet adopted in the software development industry, could be profitably used? Conversely, could programming techniques be successfully applied to filmmaking?

To answer this research question we analyzed the relevance of certain movie-production practices to software engineering and *vice versa*.

- **RQ3**: Which filmmaking practices require or actively involve some degree of cooperation? Is it possible to identify practices of cooperative work in the film industry that can also be applied to software development?

Our goal here was to highlight potential connections between these two practices, in light of this specific requirement (collaboration or team work).

2.2. Search Process

A crucial stage in any literature review is the collection, and the subsequent analysis, of the sources selected (Brereton et al., 2007; Kitchenham et al., 2009; Piper, 2013; Petersen et al., 2015; Farina et al., 2022).

We carried out a series of exploratory searches to locate benchmark papers in the field. To perform our searches we used the following electronic databases:

- Google Scholar;
- ACM Digital Library;
- IEEE Xplore;
- Science Direct;
- ReserchGate.

Our exploratory searches demonstrated a crucial lack of relevant literature on the topic. We only found one study (Graham, 2004), exploring the connection between programmers and painters, which was only mildly relevant to our investigation. For each of the research questions we introduced earlier on, we specified—in accordance with the best standards and norms of our discipline—a number of keywords characterizing them. Subsequently, we constructed relevant search queries with the aim of broadening our initial searches. An example of a query we used for the first research question follows below:

- (Management OR development OR creation process OR production) AND (practices OR techniques OR steps OR stages) AND (movie production OR filmmaking OR cinema industry) AND (software development OR software engineering OR programming) AND (similarities OR resemblance OR parallels OR correlation).

Additional examples of queries we used for the second and the third research questions can be found below:

- (Lean OR Agile) AND (practices OR techniques OR steps OR stages) AND (movie production OR filmmaking OR cinema industry)
- (Cooperation OR teamwork OR collaboration) AND (practices OR techniques) AND (movie production OR filmmaking OR cinema industry).

We built a literature log to classify, store and further analyse the results obtained for all the search queries we performed.

2.3. Literature Selection

The next step involved in our review required the application of our Inclusion and Exclusion criteria to the set of papers we gathered through our searches. The formulation of inclusion/exclusion criteria is a necessary step for any systematic literature review (Kitchenham et al., 2009). Inclusion and exclusion criteria typically help identifying relevant studies for inclusion or filtering out improper or irrelevant sources (Kitchenham, 2004). With respect to this point, we would like to notice that as the topic of our research is mostly overlooked in the relevant literature, the inclusion and exclusion criteria were formulated in a rather loose or comprehensive way; that is, they were formulated to provide enough coverage, hence a wider literature base for our study. For this reason, we decided to include in this review a broad range of papers published in software engineering in the last 20 years and put no time-constraint for including papers related to the cinema industry. Moreover, the selection process was applied primarily after the initial search, but some of the criteria (such as time frames and language restrictions), were included in the first stage as well. Therefore, a paper was selected for inclusion if it met all the following basic requirements:

- **Time**: filmmaking—no time-constrain; software development—last 20 years
- **Peer-review**: books, articles in reputable journals, and conference proceedings
- **Language**: literature written in either English or Russian. Russia is home of a large and very productive IT community. In addition, the Russian language is spoken by ∼300 millions people worldwide and some of the researchers who

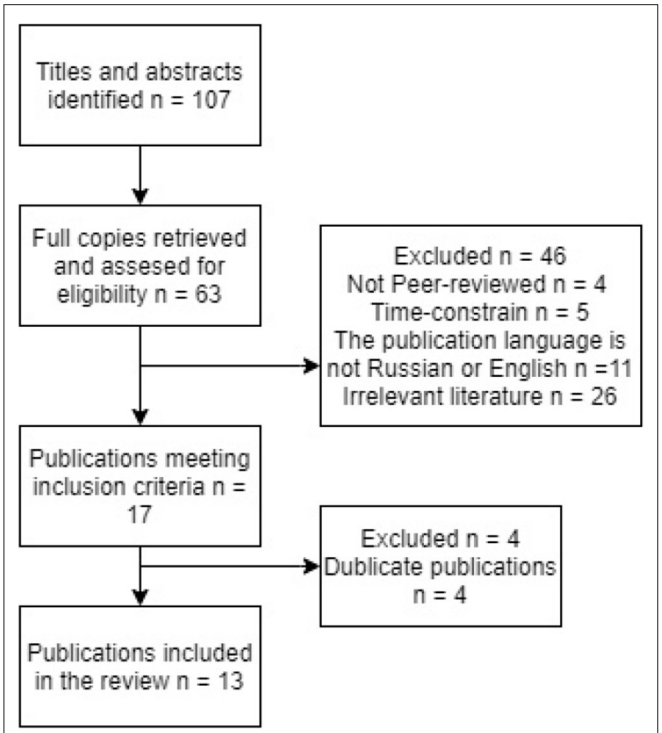

FIGURE 1 | Flow chart diagram.

participated in this study are Russians. Hence, we believe it is fine, also for cross-cultural reasons, to consider this language requirement in our review.

- **Relevancy**: literature relevant to the chosen topic.

On the contrary a paper was excluded from our log and hence from this study if it did not met any of the inclusion criteria above-mentioned or if it was a duplicate. The process described above is summarized in **Figure 1** below.

We would like to note that all the studies that were included in the review were cross-checked for adherence to the above-mentioned inclusion criteria by all the researchers involved in the study.

2.4. Quality Assessment

Having specified the search process as well as the Inclusion/Exclusion criteria we adopted in our research protocol, we next focused on the qualitative assessment of our results.

In order to maximize objectivity and minimize the possibility of subjective interpretations and/or mistakes, the studies we selected were simultaneously assessed by all the researchers involved in the review process.

To determine the scientific reliability and accuracy of the studies included, we first generated a set of questions that we applied to each of the papers included in our review. We then assigned to each question a specific score, so that papers' quality could be objectively assessed.

If "yes" was given as an answer, then the paper would receive 1 point on our scoring board. If "partially" was given as an answer, then the paper would receive 0.5 points on our scoring board. If "no" was given as an answer, then the paper would receive 0 points on our scoring board.

The total score for each paper was then calculated and a qualitative assessment performed. The results of this process can be observed on **Tables 2**, **3** below. Below, we also report the set of questions we used in order to evaluate the papers' quality and to establish their scientific soundness:

1. Were the objectives of the paper clearly stated?
   - 1 point if the objectives were explicitly stated;
   - 0.5 points if the objectives were clear enough to be understood;
   - 0 points if no objectives were stated, or if the objectives were hard to determine, or if they didn't relate to the proposed research.
2. Did the paper achieve the objectives stated?
   - 1 point if the paper achieved the proposed objectives or goals or if it implemented a solution that was instrumental to such an end;
   - 0.5 points if the paper achieved the proposed goals with some limitations, deviations, or shortcomings;
   - 0 points if the paper did not achieve the proposed goals or objectives.
3. Was the research process clearly described? In other words, was it transparent and reproducible?
   - 1 point if the paper clearly specified methods, technologies, and data, along with all necessary references and sources needed to reproduce the research;
   - 0.5 points if minor details were lacking or if the research process was not fully transparent;
   - 0 points if it was impossible to restore the sequence of actions (hence the research process was not transparent), or if certain critical details were missing.
4. Were the results properly evaluated?
   - 1 point if the paper provided a clear and systematic analysis of the results; hence if the results were assessed in an appropriate manner;
   - 0.5 points if the paper offered only a partial analysis of the results or if they could have been better analyzed;
   - 0 points if the results provided were not sufficiently scrutinized or if there was no attempt to evaluate the data.
5. Was the conclusion sound?
   - 1 point if the conclusion was deemed to be logically sound and scientifically grounded;
   - 0.5 points if the conclusion was acceptable but some limitations were found;
   - 0 points if the conclusion was overstated or if certain techniques (such as spin) were deployed.

Thus, the total score that could potentially be attributed to each paper ranged from 0 to 5. The results of this quality assessment procedure can be found on **Tables 2**, **3** below. This process of

TABLE 2 | Quality assessment—statistics.

QA score	Quantity	Percentage
0–2.5	0	0
2.5–4	5	38.4
4–5	8	61.6

TABLE 3 | Quality mean values by year.

Years	Quantity	Average quality
1983–2004	4	4.5
2013–2015	4	3.5
2017–2020	5	4

quality assessment was performed on the 13 papers selected for inclusion in our review and it was, as noted above, of paramount importance to evaluate the quality of the papers we included in this study. As the tables below clearly demonstrate, the vast majority of the papers we ended up including in our study were of either good quality (overall score between 2.5 and 4) or of high quality (overall score between 4 and 5). On these grounds, we can infer that the results we gathered from the papers we included in our log are accurate and scientifically sound.

3. RESULTS

Before we go on to cluster the papers we selected in our study by topics, we quickly show their distribution by year of publication (**Figure 2** below). This data is meant to provide our readers with some general, yet important information about how research on this topic has developed in recent years.

Table 4 below clusters the papers we selected around two main topics: (i) filmmaking practices and (ii) analogies between filmmaking practices and software development. **Table 5** below shows whether our selected papers belonged to gray or white literature. **Figure 3** below further specifies the distribution (in terms of percentage) of the papers we selected by publisher. As customary for a literature review, we focused—mostly—on white literature and on secondary sources, even though we also included in our study some gray literature. We would like to note here that including gray literature in a literature review (so advocating a multivocal approach to literature reviews) is becoming an increasingly acceptable practice in software engineering (Mahood et al., 2014; Garousi et al., 2016, 2020).

The first topic we analyzed (movie practices) is characterized by a relatively large amount of publications. Therefore, we arranged the relevant literature into two main streams or parts: (i) papers describing the movie creation process and (ii) papers describing the techniques used, in filmmaking, by famous movie directors. If, for the first topic, inclusion was pretty straightforward, the selection process for the second topic proved to be slightly more complicated. More specifically, it was necessary to exclude the sources that did not explicitly relate to the research (for example, papers about artistic, or non-management techniques).

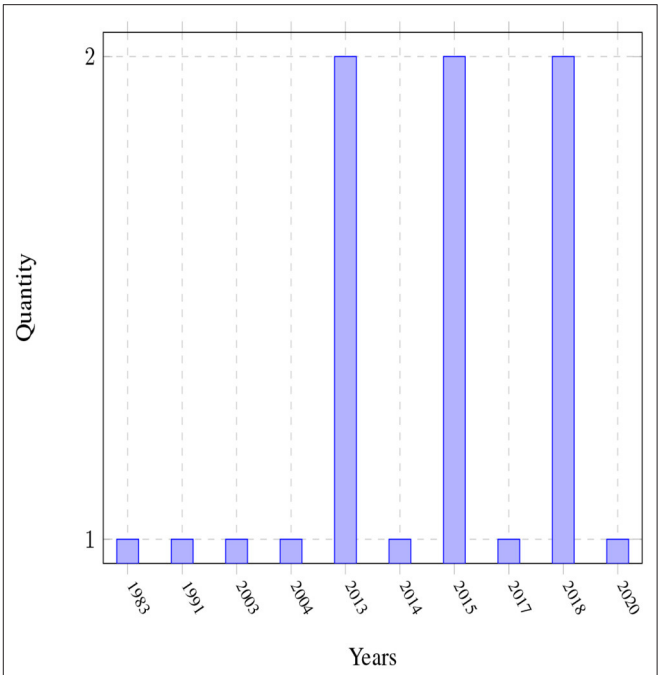

FIGURE 2 | Paper distribution by year of publication.

TABLE 4 | Number of sources per topic.

Searched topics		Initial sources		Included sources	
		White	Gray	White	Gray
Filmmaking practices	The movie creation process	19	0	1	0
	Directors' techniques	12	9	4	0
Resemblance	Similarities between creativity and programming	1	4	1	2
	Lean and Agile in cinema	1	17	1	4

The second topic instantiated parallels between filmmaking and software development. We first examined the similarities between certain creative processes and particular engineering or programming techniques. We only found a few sources attesting to some kind of resemblance between filmmaking and software production. Among such sources, we came across two articles (gray literature, Garousi et al., 2019) that were proposing an interesting parallelism between movie management and software development. We also explored the possibility to apply software development approaches (such as Lean and Agile principles or methodologies) to filmmaking. Most of the sources we found here could be classified as gray literature; however, there were some books describing curious experiments, for example, about Lean

filmmaking (more on this below), that we ended up including in our review.

Having presented our research protocol, we next focused on answering the research questions characterizing our study, in light of our findings.

3.1. About RQ1: Resemblance Between Filmmaking and Software Development

To answer this research question, we preliminary analyzed studies describing some analogies between art and software development. As mentioned above, we found an interesting study proposing some similarities between the work of software engineers (programmers) and the work of painters (Graham, 2004). More specifically, this study considered both artists and programmers as "makers." The analogy thus revolved around the fact that both artists and programmers produce things, that can be practically used or appreciated, by different people. A second important analogy described in the study, which is partly derived from the first one we discussed above, concerns the idea that both artists and programmers are ultimately creative individuals (Sacks, 1992), deeply involved in acts of creativity (Csikszentmihalyi, 1990, 2015; Greenberg, 2007), that shape and mould the world around themselves (Csikszentmihalyi, 2014).

Other potential analogies or parallels were observed. In particular, a paper by Calvo (2013) describes the structural similarities between filmmaking and the process involved in the production of any given software. The first step in the production of a movie is known as script writing (or screenwriting, Parker, 1999). This is a process that requires writing down the movement, actions, expression and dialogue of the characters in the screenplay (Field, 2008). Screenwriting serves to construct the film's narrative and to shape it and design it to match the targeted audience (Hueth, 2019). The corresponding process in software development involves the analysis and design of software parts. In software engineering, this step typically includes the formation of a core structure as well as the identification of functional and non-functional requirements and explanation of user interactions (Pressman, 2005). Thus, there seems to be a relatively strong parallel (in terms of structures at least) between the processes involved in the production of a movie and those characterizing the development of any given software system architectures.

Another important analogy concerns the similarity between screenwriting and code production. Filmmakers make storyboards, blueprints for their movie, and these are typically created on the basis of a script base (Van der Lelie, 2006). Creating project prototypes capable of informing product development is a general technique used in software development as well (Winkler et al., 2013; Huber et al., 2020).

TABLE 5 | Types of sources selected.

Grey literature
Magazines
- Code
- Filmmaker
Blogs
- Pipefy
- The Beat
- Axosoft Dev
- Carey Martell
White literature
Journals
- Essays
- Extreme Leadership: Leaders, Teams, and Situations Outside the Norm
Conferences
- Conference and Symposium on the Foundations of Software Engineering
Books
- O'Reilly Media, Inc.
- National Academies Press Washington, DC
- Gulf Professional Publishing
- Vintage
- Springer
- Cambridge Scholars Publishing

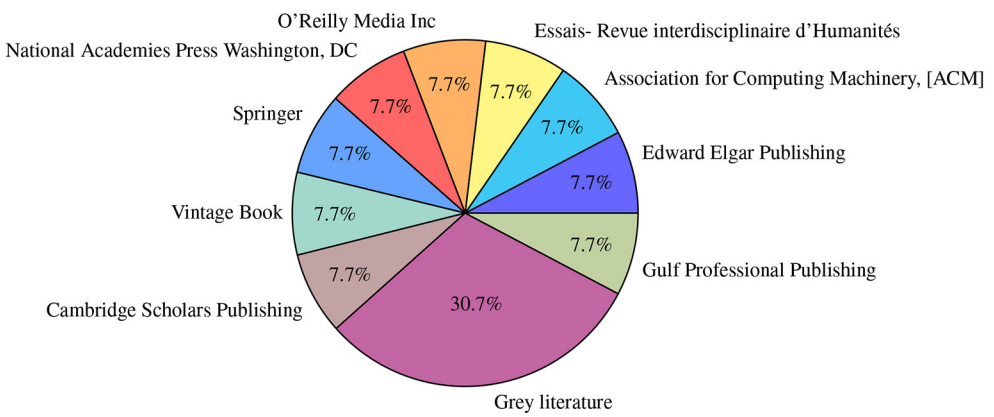

FIGURE 3 | Papers distribution—by publisher.

Yet, we found that there is no direct and precise analogue of the screenplay editing process in software engineering (Millard, 2010; Koivumki, 2011). Screenplay editing processes typically require multiple re-shoots, test screenings, and constant honing. Nevertheless, it is, of course, possible to argue that honing is crucially important in software development, inasmuch as a solution to problems often requires multiple interactions and progressive, incremental changes.

Another significant point of contact between these two practices is the fact that both are designed and are indeed intended for a human audience (Altenloh, 1914; Dijkstra, 1979; Seffah et al., 2005; Hanich, 2018). Both the filmmaker and the software developer in organizing their work, must keep in mind the user; hence, they must make their products (be it movies or programs) easily accessible by virtually anyone.

3.2. About RQ2: Potentially Overlapping Practices Used in Filmmaking and Software Development Techniques

The second research question was split in two different parts: (i) potential application of filmmaking techniques to software development and (ii) potential usage of programming approaches in the movie industry.

3.2.1. Potential Adaptation of Filmmaking Techniques to Programming

Unfortunately, no researcher has been investigating this issue in the scholarly community at the time of writing; hence, there is an instructive lack of studies with respect to this point. To solve this issue and to provide some basis for our review, we decided to review some approaches and techniques used by various movie directors with respect to management practices.

Akira Kurosawa, a famous Japanese director, shared his vision and thoughts about filmmaking in his autobiography (Kurosawa, 1983). In this book, he is portrayed as actively involved and directly participating in every single aspect of the movie production (including writing the scripts, developing the design, preparing the actors, placing shots and editing; Stiglegger, 2001). This, it can be argued, is what gave Kurosawa's movies the feeling that the director was in command over every aspect of the production process (Richie and Mellen, 1998) as well as what determined part of their success. This capacity, we note, is also important in software engineering, where having a leader, who is highly experienced in every aspect of the project development, is an important condition for the successful implementation of the project itself (Alefari et al., 2017).

However, Kurosawa was not solely responsible for the writing of his movies. He was actively collaborating with other screenwriters. Each of them worked on the same script, often at the same time, adding text and suggestions and often different points of view. At a later stage, Kurosawa reviewed the scripts produced and picked what he thought was the best, among those that were submitted to him. This rather authoritative way of selecting and organizing contents helped Kurosawa streamlining potential disagreement among different scriptwriters. We note that this approach is often replicated in the software development industry. For example, the process of architectural design is in many ways analogous to that of screenwriting. In such a process, each qualified team member does provide his or her own version of the project structure, but then the project leader intervenes and ultimately chooses which one he would like to pursue.

However, it is worth also noting that despite this rather authoritative approach to production during his career, Kurosawa preferred working with the same group of creative technicians, crew members and actors, popularly known as the "Kurosawa-gumi" or Kurosawa's group. The intimate relations between professional team members usually contributes to increase the project's quality. It can surely be argued that Kurosawa-gumi ensured the success of Kurosawas'movies (Nogami, 2006). Again, the same results (project's improvements)—we notice—are often achieved by following this practice in software engineering. Well-established and long-term collaborative partnership between and within software engineering teams do provide higher work efficiency during the project development and contribute to maximize outcomes as well as results (Whitehead, 2007).

Another very famous director—Steven Spielberg—who is also a fan of Kurosawa—perhaps inspired by him—developed his own way to organize and supervise the production of a movie (Acuña, 2018). One of the greatest strengths underlying Spielberg's work is arguably the capacity to meticulously plan the pre-production stage of his movies (Gordon, 2007). This involves paying a lot of attention to the movie script as well as to the storyboards and to the mockup, a full-sized structural model of the movie typically used for reviewing format (Awalt, 2014). Such careful planning, it has been argued helped Spielberg sticking to deadlines and production budgets, determining many of his successes (Morris, 2007). On a similar vein, we note that the waterfall approach in software engineering, which typically allows the breakdown of projects into linear sequential phases, each characterized by a specific deliverable that depends on the deliverables of the previous phase and corresponds to a specialization of tasks (Adenowo and Adenowo, 2013), requires a similar approach.

Another decisive component, which arguably contributed to the success of Spielberg's work is his direct involvement in the selection of the professionals used in the movie production (Awalt, 2014). Spielberg is known to give special attention to friendly management and social skills. His priority is to achieve the maximum degree of cooperation possible in any given project in which he is involved. This requires remaining open to suggestions for improvement coming from other team members (Peña-Acuña, 2018). As software development is a deeply collaborative process (Saeki, 1995; O'Neill, 2001), we notice that successful team leaders in software engineering often adopt this type of practice (Whitehead, 2007).

3.2.2. Potential Usage of Programming Techniques in the Movie Industry

To answer this second sub-question we looked at existing successful implementation of Lean principles in filmmaking.

First, we found that the Lean principle of focusing on people, its motivation and collaboration (Poppendieck and Cusumano, 2012) has been largely adopted by moviemakers. Filmmaking is

an quintessentially collaborative practice that relies on different types of talents, skills, and expertise, all working collectively to successfully produce the end product, the movie (Hodge, 2009). Pixar studio recently introduced a policy that allows everyone in the company, irrespective of their position, to express their opinions, provide feedback, and share their ideas with the director. The aim of this policy is to maximize collaborative interactions among team members and boost production and efficiency (Moreira, 2020).

Besides, it is worth noting that the Pixar studio represents a very good illustration of how a company can successfully use lean principles. At various stages of the film production, their workers actively use various lean tools and procedures (such as Kanban and Andon) to minimize waste and avoid production bottlenecks. For example, in his movies Pixar director, Jason Blum, aims at reducing waste by producing small-budget films, which use relatively inexpensive shots and limit the number of locations, dialogues, and speaking roles (Yakimchuk, 2017).

Another important thing to mention in this context is the first Lean filmmaking experiment made by Eddy and Eddy (2020). The goal of this experiment was to apply a number of Lean principles to the movie production process. In his experiment, which involved recording a hackathon, groups of eight-nine people were creating films for 2 days working in abbreviated cycles. Each film cycle was showed to an audience with the purpose of gathering feedback, which informed subsequent changes in the production line. In this way, the need for a script was abolished, as it was considered redundant. The experiment was quite successful and the authors' achievements confirmed the possibilities to tailor and adapt software engineering techniques to different contexts (such as movie production).

To further analyze the potential relevance of software engineering methods, practices, and techniques to filmmaking, we also looked, as mentioned in the introduction, at the Agile approach. Specifically, we analyzed it in terms of movie production. A study by Figueroa (2015) claimed that Agile methodologies can be applied successfully in filmmaking based on similarities between movies and software projects. According to Martell (2015), Scrum methods can be applied to any kind of work and are "perfect systems for adapting to the film industry" Martell (2015).

3.3. RQ3: Collaborative Practices in Filmmaking and Software Development

3.3.1. Teamwork and Leadership in Filmmaking

Another important aspect of our research involved an analysis of the importance of collaborative efforts. Such an analysis was instrumental to answer the third research question characterizing our work.

First, we researched teamwork practices in Stanley Kubrick's work (Perko, 2018). We found out that one the many skills in which Kubrick excelled was communication (Falsetto, 2001). Not only that, but we understood that he was always open to suggestions and ideas from other crew members and that he was also a good listener (Phillips, 2013). It can thus be argued that this peculiar talent and overall attitude also helped him winning over his people hearts and minds.

Our investigation of Kubrick's work revealed that Kubrick, in order to streamline movie production, often used to divide larger tasks into smaller ones. More specifically, we found out that he used to separate crew members in smaller groups and assign them a team leader (Falsetto, 2001). This approach was pursued to determine a fair and efficient division of the tasks among workers (Adam, 2016). Moreover, such an approach to team management was instrumental to introduce another crucial cooperative practice; that of mentorship. Team leaders and even Kubrick himself occasionally assumed the role of mentors to guide, orient, and help young and inexperienced film workers. This was instrumental to provide them with opportunities for learning and developing their respective careers.

However, it is important to note that there wasn't an unlimited degree of freedom for collaborative team members under Kubrick's supervision. Krubrick extensively used the social capital as well as the connections he had accumulated over his career, to persuade people, put under pressure, or even directly control them during the production process of his movies Kubrick (2001).

A number of studies mention leadership as a crucially important factor in collaborative processes (Ferren and Stanton, 2004; Slater, 2005; Ciancarini et al., 2021a,b). For example, Mainemelis and Epitropaki (2014) examined the way in which Francis Ford Coppola organized the production of his masterpiece "The Godfather." The director in this case used an approach that can be called of extreme leadership, according to which "the leader act as the troublemaker, who induces crises and creates chaos" (Mainemelis and Epitropaki, 2014).

During the shooting of The Godfather, Coppola used to continuously change script as well as the shooting process. Coppola had understood that film creation process is fluid and one must always search for new ideas and opportunities to improve the project. This flexible approach allowed him to better integrate creative contributions from its crew mates.

Now, we know that at the beginning the production of The Godfather wasn't going so well because of these continuous changes. We also know that Coppola was on the verge of getting fired; yet he continued applying his style to the production of his movie, making unusual and risky decisions (such as the decision to cast Brando and Pacino for leading roles). Nevertheless, Coppola efforts were crowned with success, and the movie has given him international acclaim.

3.3.2. Correlation Between Collaboration Practices in Movie and Software

As the reader may recall the significance of the third research question lied in the identification of movie practices that are potentially applicable to software development. At the time of writing we can not say whether the techniques we discussed above can necessarily be beneficial in the software industry. This is because the current available scientific literature did not investigate this specific question.

However, our analysis was significant because we discovered that software development and filmmaking share a common

nature; that of both being highly collaborative and creative social practices (Mitchell et al., 2003). Collaboration in software engineering and filmmaking is certainly complex and happening at many different levels, as we have seen above. For example, project members in both the IT and Cinema Industries typically deal with how the work is separated, how the results are gathered and combined, how the resources are allocated, or how the communication is organized. This requires a significant amount of coordination, cooperation and organization; however, it seems that the combination of these factors (and possibly of more, involving of course also a good degree of creativity) is a recipe for high work performance and success. On these grounds, we can probably suggest that, at least, some filmmaking practices and techniques could be successfully adopted in software development.

4. SUMMARY OF OUR RESULTS

In this section, we schematically summarize for the reader what we have achieved so far.

4.1. Analogies Between Software Development and Filmmaking (RQ1)

We identified a series of important commonalities between fimmaking and software engineering, such as:

- both artists and programmers can be seen as *makers*.
- both artists and programmers can be seen as highly creative individuals.
- both filmmaking and software development require and demand continuous cooperative interactions and incremental changes.
- both software development and filmmaking are designed are indeed intended for a human audience.

4.1.1. Movie and Software Development Techniques (RQ2)

Our review of the literature allowed us to identify relevant information about the adaptation of software development techniques in the movie industry: Our literature review, in particular, established that:

- this topic is not yet well-investigated in the relevant scientific community, hence it showed the importance of our selected topic for future research in the field.
- Kurosawa used to control and direct almost every aspect of his film production. Our work also demonstrated that this capacity of leadership is crucially important in software engineering.
- Kurosawa, while centralizing decision making, was also seeking collaborative interactions with his team members in the production of the screenplay and that he preferred working with the same group of creative collaborators (or gumi). Our work also showed that long-term team engagement is crucially important in the software development industry.
- Spielberg's work is an excellent example of Waterfall.
- Spielberg pays maniacal attention to management, cooperation, and social skills. Our review also showed the centrality of such an open minded, collaborative approach to software engineering/development.

We nevertheless also found that software development techniques or approaches can also be used in filmmaking. In particular, we found:

- that the lean principle of focusing on people, its motivation and collaboration is adopted by moviemakers.
- that Pixar studio successfully uses lean principles.
- that Agile methodologies and especially Scrum can be also successfully applied in film industry.

4.1.2. Collaborative Filmmaking Practices (RQ3)

In addition, in the context of filmmaking practices we also found:

- that openness to suggestions and ideas from other crew members is widely used.
- that division of larger tasks into smaller ones is often beneficial.
- that mentorship and supervision are profitable.
- that leadership is an important factor in collaborative processes as well as in the implementation of any project's plan.

5. LIMITATIONS AND THREATS TO VALIDITY

In this section, we critically analyse the above-mentioned results, focusing especially on potential biases in selection as well as on other sort of limitations, and on potential threats to the validity of this study (Keele et al., 2007; Akl et al., 2019).

5.1. Results Validation

Results validation is a significant part of any literature review (Robson and McCartan, 2016). We preliminary reviewed the outcomes of this study through self-assessment. In particular, we investigated the risk of bias in our study. **Table 6** describes the possible biases potentially affecting our study and the solutions adopted to avoid them.

Although it is, de facto, impossible to perform a fully unbiased research; we aimed at reducing biases as much as possible. Based on the explanation given on **Table 4** below, we reached the conclusion that our level of bias is moderate, and—as a consequence—that our work can be considered to be scientifically reliable.

5.2. General Limitations

During the review process, we faced some challenges, which gave us an opportunity to notice the many potential limitations affecting this research. The biggest limitation affecting this study probably concerns the lack of relevant academic literature connecting the practices of software development and movie production and the limited amount of studies included in this review. As a consequence, the relationship between the practices underlying software development and film-making are investigated by a very small number of papers in scientific databases and our claim partially lack a strong scientific backup. Another important limitation affecting our study, which directly descend from the first one we mentioned above, is the heavy

TABLE 6 | Potential biases.

Type of bias	Comment
Pre-trial bias	
Flawed study design	• The Methodology section clearly describes objectives, research questions, plan and methods for the review. So, this bias type is minimized and mostly avoided.
Selection bias	• We adopted rigorous criteria for literature selection and that—we believe helped avoiding this type of bias. Section 2 above describes this point in more details.
Bias during trial	
Data collection bias	• A bias (involving conflict of interest, for example) may occur in this stage. We would like to notice that we do not have any conflict of interest to declare. Hence, there was no bias relationship between the authors of the paper and the authors of the sources selected for the review. Another bias typically occurring in the stage of data collection concerns the inclusion of gray literature in the review. We admit that we included a lot of gray literature in this study and that might be problematic. However, we also reiterate—as argued above—that including gray literature in literature reviews is becoming an increasingly acceptable practice in software engineering. We would also like to notice that while we found many scientific articles describing both the principles and the background for software and filmmaking practices, we couldn't locate any information about their resemblance in peer reviewed manuscript; hence, we had to include sources belonging to gray literature.
Bias after trial	
Analysis bias	• Despite this work tries to find confirmation for our hypothesis; the bias is still moderate because there is not much literature that researched our topic.
Publication bias	• This type of bias normally occurs in published academic research, especially when certain research influences the decision to publish, distribute or select for publication a given paper Begg, 1994 We followed the PRISMA checklist and specified our research protocol upfront. Therefore, we are confident that we did not incur in this sort of bias.

TABLE 7 | Prisma checklist.

Section	Reported on page
TITLE	
Title	1
ABSTRACT	
Structured summary	1
INTRODUCTION	
Rationale	1
Objectives	1-2
METHODS	
Protocol and registration	2
Eligibility criteria	3
Information sources	3
Search	3
Study selection	3
Data collection process	3
Data items	3
Risk of bias in individual studies	3
Summary measures	4-5
Synthesis of results	1
Risk of bias across studies	1
Additional analyses	1
RESULTS	
Study selection	3-4
Study characteristics	4
Risk of bias within studies	4
Results of individual studies	4
Synthesis of results	4-7
Risk of bias across studies	4
Additional analysis	7
DISCUSSION	
Summary of evidence	7-8
Limitations	7
Conclusions	8
FUNDING	
Funding	9-10

reliance on gray literature. There is, potentially, a lot of useful scientific information presented in reports, blogs, and various other non-academic venues (Paez, 2017); however, when using it, one must be particularly careful, so as to avoid relying on incorrect or inaccurate data. We carefully scrutinized such literature in accordance with the best practices of our discipline (Schmucker et al., 2013; Borrego et al., 2014), and attempted to include only reliable data in our review.

5.3. Critical Assessment Against Benchmark Questions

A final step in the evaluation of research findings involves their critical assessment against a set of benchmark questions, which can be used as a point of reference to assess the overall quality of a literature review (Kitchenham, 2004):

1. *Are the review's inclusion and exclusion criteria described and appropriate?* All criteria used for inclusion or exclusion were mentioned upfront in our research protocol. All the criteria used in this research seem to be reasonable and relevant to the topic. We thus believe that they are appropriate and coherent for this research.
2. *Is the literature search likely to have covered all relevant studies?* The process we established to gather papers *via* search queries as well as the methods we used to search relevant databases were sound and comprehensive. We are thus confident that our literature searches were representative and accurate.
3. *Did the reviewers assess the quality/validity of the included studies?* We count this condition as sufficiently met as we included in our review a brief quality assessment of the findings included in this study.
4. *Were the basic data/studies adequately described?* We believe this condition is met because we built a reading log to put all the relevant information extracted from the papers we

selected. This allowed us to process our data systematically and comprehensively.

5.4. Adherence to PRISMA Checklist

In order to produce a systematic and coherent piece of work, as noticed in Section 2 above, our research protocol was developed in accordance with the PRISMA checklist. **Table 7** describes in some detail how and where we dealt with the most important items of the PRISMA checklist. This checklist demonstrates the quality of the review and allows our readers to more comprehensively evaluate its strengths and weaknesses:

6. CONCLUSIONS

Naturally, the analysis performed on our findings is somehow subjective, hence subject to interpretations, which may lead to disagreement. However, as mentioned above, in order to maximize objectivity and reduce suggestiveness, all authors of this research were directly involved in the analysis.

Our analysis demonstrated that software development and filmmaking indeed have a common nature. This study also showed that these two seemingly different processes are not only highly creative practices, but also that they are cooperative in character. We managed to show that artists and programmers can be considered as makers of things and that they are—to some extent at least—moved by the same goal; namely making things for a human audience. In addition, our study demonstrated—through a comparative analysis—, that it is possible to use and successfully deploy the methods of cooperative work used in the film industry in software development (Callens, 2013). Similarly, we showed that some methodologies used in software engineering (such as Agile and Lean) can be profitably adapted by filmmakers.

This review therefore suggested a new promising line of research for the exploration of the relations between software engineering and filmmaking. However, we are aware that this work constitutes only a preliminary and rather partial attempt of analysis; it is nevertheless hoped that this review will broaden interest in this topic and provide new theoretical grounds for more detailed explorations into these extraordinarily rich and fascinating set of phenomena. As a matter of fact, we will now turn our attention to performing an empirical analysis on the field. The goal of this analysis will be to articulate and specify the findings of this review, by considering whether and how the variety in genres of movies (action, comedy, drama, science fiction, . . .) can be mapped to the variety of software being produced (apps, real time, embedded, commercial, etc. . . .).

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

All authors: conception, elaboration, and review. All authors contributed to the article and approved the submitted version.

FUNDING

The authors thank Russian Science Foundation for generously supporting this research with grant number 22-21-00494.

REFERENCES

Acu na, B. (2018). *Understanding Steven Spielberg*. New Horizon. Cambridge Scholars Publishing.

Adam, S. (2016). *The Wealth of Nations*. Toronto, ON: Aegitas.

Adenowo, A. A., and Adenowo, B. A. (2013). Software engineering methodologies: a review of the waterfall model and object-oriented approach. *Int. J. Sci. Eng. Res.* 4, 427–434. Available online at: https://www.ijser.org/paper/Software-Engineering-Methodologies-A-Review-of-the-Waterfall-Model-and-ObjectOriented-Approach.html

Agrawal, M. (2016). *Filmmaking: A Project Management Case Study for Software Development*.

Ahmed, S. U., Jaccheri, L., Sindre, G., and Trifonova, A. (2009). "Conceptual framework for the intersection of software and art," in *Handbook of Research on Computational Arts and Creative Informatics* eds J. Braman, G. Vincenti, G. Trajkovski (Hershey, PA: IGI Global), 26–44. doi: 10.4018/978-1-60566-352-4.ch002

Akl, E., Altman, D., Aluko, P., Askie, L., Beaton, D., Berlin, J., et al. (2019). *Cochrane Handbook for Systematic Reviews of Interventions*. Hoboken, NJ: John Wiley & Sons.

Alefari, M., Salonitis, K., and Xu, Y. (2017). The role of leadership in implementing lean manufacturing. *Proc. Cirp* 63, 756–761. doi: 10.1016/j.procir.2017.03.169

Altenloh, E. (1914). A sociology of the cinema: the audience. *Screen* 42, 249–293. doi: 10.1093/screen/42.3.249

Awalt, S. (2014). *Steven Spielberg and Duel: The Making of a Film Career*. Rowman & Littlefield.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). *The Agile Manifesto*. Available online at: https://agilemanifesto.org/

Begg, C. B. (1994). "Publication bias," in *The Handbook of Research Synthesis*, eds H. Cooper and L. V. Hedges (New York, NY: Russell Sage Foundation), 299–409.

Bond, G. W. (2005). Software as art. *Commun. ACM* 48, 118–124. doi: 10.1145/1076211.1076215

Borrego, M., Foster, M. J., and Froyd, J. E. (2014). Systematic literature reviews in engineering education and other developing interdisciplinary fields. *J. Eng. Educ.* 103, 45–76. doi: 10.1002/jee.20038

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* 80, 571–583. doi: 10.1016/j.jss.2006.07.009

Callens, B. (2013). *Pixar's 22 Rules of Storytelling and How They Apply to Software Development*. Available online at: https://wraltechwire.com/2013/09/26/pixars-22-rules-of-storytelling-and-how-they-apply-to-software-development-13927/

Calvo, A. (2013). *Why Filmmaking is Like Software Design*. Available online at: https://filmmakermagazine.com/67052-why-filmmaking-is-like-software-design/#.Ylb6lMhBw2w

Ciancarini, P., Farina, M., Masyagin, S., Succi, G., Yermolaieva, S., and Zagvozkina, N. (2021a). Non verbal communication in software engineering-an empirical study. *IEEE Access* 9, 71942–71953. doi: 10.1109/ACCESS.2021.3075983

Ciancarini, P., Farina, M., Masyagin, S., Succi, G., Yermolaieva, S., and Zagvozkina, N. (2021b). Root causes of interaction issues in agile software development teams-status and perspectives. *Adv. Intell. Syst. Comput.* 2, 1017–1036. doi: 10.1007/978-3-030-73103-8_74

Cramer, F., and Gabriel, U. (2001). Lanham, MD: Software art.

Csikszentmihalyi, M. (1990). *The Domain of Creativity*. New York City, NYC: Harper.

Csikszentmihalyi, M. (2014). "Society, culture, and person: a systems view of creativity," in *The Systems Model of Creativity* (Springer), 47–61. doi: 10.1007/978-94-017-9085-7_4

Csikszentmihalyi, M. (2015). *The Systems Model of Creativity: The Collected Works of Mihaly Csikszentmihalyi*. Springer. doi: 10.1007/978-94-017-9085-7

Dijkstra, E. (1979). "Programming considered as a human activity," in *Classics in Software Engineering* (Yourdon Press), 1–9.

Dingsøyr, T., Dybå, T., and Moe, N. B. (2010). "Agile software development: an introduction and overview," in *Agile Software Development* eds T. Dingsøyr, T. Dybå, N. Moe (Berlin: Springer), 1–13. doi: 10.1007/978-3-642-12575-1_1

Donelan, J. (2007). Lessons in filmmaking. *Comput. Graph. World* 34–39.

Ebert, C., Abrahamsson, P., and Oza, N. (2012). Lean software development. *IEEE Comput. Archit. Lett.* 29, 22–25. doi: 10.1109/MS.2012.116

Eddy, D., and Eddy, K. (2020). *The Art of Lean Filmmaking: An Unconventional Guide to Creating Independent Feature Films*. Lean Film making.

Falsetto, M. (2001). *Stanley Kubrick: A Narrative and Stylistic Analysis*. Berlin: Greenwood Publishing Group.

Farina, M., Gorb, A., Kruglov, A., and Succi, G. (2022). Technologies for GQM-based metrics recommender systems: a systematic literature review. *IEEE Access* 10, 23098–23111. doi: 10.1109/ACCESS.2022.3152397

Feld, W. M. (2000). *Lean Manufacturing: Tools, Techniques, and How To Use Them*. Westport, CT: CRC Press. doi: 10.1201/9781420025538

Ferren, A. S., and Stanton, W. W. (2004). *Leadership Through Collaboration: The Role of the Chief Academic Officer*. Boca Raton, FL: Greenwood Publishing Group.

Field, S. (2008). *The Definitive Guide to Screenwriting*. Westport, CT: Random House.

Figueroa, G. (2015). *Lights, Camera, Software Development!*. Available online at: https://www.projecttimes.com/articles/lights-camera-software/

Fishwick, O. P., Malina, R., Sommerer, C., Bertelsen, W., and Fishwick, P. (2003). Aesthetic computing "manifesto". *Leonardo*. 36:255. doi: 10.1162/002409403322258556

Fishwick, P. A. (2008). *Aesthetic Computing*. New York City, NY: MIT Press.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2016). "The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering* (Cambridge, MA:), 1–6. doi: 10.1145/2915970.2916008

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Inform. Softw. Technol.* 106, 101–121. doi: 10.1016/j.infsof.2018.09.006

Garousi, V., Felderer, M., Mäntylä, M. V., and Rainer, A. (2020). "Benefitting from the grey literature in software engineering research," in *Contemporary Empirical Methods in Software Engineering* eds M. Felderer, G. Travassos (Berlin: Springer), 385–413. doi: 10.1007/978-3-030-32489-6_14

Gordon, A. M. (2007). *Empire of Dreams: The Science Fiction and Fantasy Films of Steven Spielberg*. Limerick: Rowman & Littlefield Publishers.

Graham, P. (2004). *Hackers & Painters: Big Ideas From the Computer Age*. Lanham, MD: O'Reilly Media, Inc.

Greenberg, I. (2007). *Processing: Creative Coding and Computational Art*. Newton, MA: Apress.

Hanich, J. (2018). *Audience Effect: On the Collective Cinema Experience*. New York City, NYC: Edinburgh University Press. doi: 10.1515/9781474414968

Hodge, C. (2009). Film collaboration and creative conflict. *J. Film Video* 61, 18–30. doi: 10.1353/jfv.0.0020

Huber, T. L., Winkler, M. A., Dibbern, J., and Brown, C. V. (2020). The use of prototypes to bridge knowledge boundaries in agile software development. *Inform. Syst. J.* 30, 270–294. doi: 10.1111/isj.12261

Hueth, A. C. (2019). *Scriptwriting for Film, Television and New Media*. Edinburgh: Routledge. doi: 10.4324/9780429461361

Janes, A. and Succi, G. (2014). *Lean Software Development in Action*. London: Springer. doi: 10.1007/978-3-642-00503-9

Kanaan, G. (2016). *Great Teamwork Makes Great Films, So What Makes Great Teamwork*? THE [LEGAL] ARTIST.

Keele, S., et al. (2007). *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical report, Citeseer.

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews, Vol. 33*. Keele: Keele University, 1–26.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering-a systematic literature review. *Inform. Softw. Technol.* 51, 7–15. doi: 10.1016/j.infsof.2008.09.009

Knuth, D. (2011). The art of programming. *ITNow* (Berlin) 53, 18–19. doi: 10.1093/itnow/bwr021

Knuth, D. E. (1984). Literate programming. *Comput. J.* 27, 97–111. doi: 10.1093/comjnl/27.2.97

Knuth, D. E. (1997). *The Art of Computer Programming, Vol. 3*. Pearson Education.

Koivumki, M.-R. (2011). The aesthetic independence of the screenplay. *J. Screenwrit.* 2, 25–40. doi: 10.1386/josc.2.1.25_1

Kubrick, S. (2001). *Stanley Kubrick: Interviews*. University Press of Mississippi.

Kurosawa, A. (1983). *Something Like an Autobiography*. New York City, NYC: Vintage.

Mahood, Q., Van Eerd, D., and Irvin, E. (2014). Searching for grey literature for systematic reviews: challenges and benefits. *Res. Synth. Methods* 5, 221–234. doi: 10.1002/jrsm.1106

Mainemelis, C., and Epitropaki, O. (2014). Extreme leadership as creative leadership: reflections on francis ford coppola in the godfather. *Extreme Leadersh.* 187–200. doi: 10.4337/9781781002124.00024

Martell, C. (2015). *Agile SCRUM for Film-makers: How to Produce Movies & TV Shows in Half the Time*. Martell Books.

Millard, K. (2010). After the typewriter: the screenplay in a digital era. *J. Screenwrit.* 1, 11–25. doi: 10.1386/josc.1.1.11/1

Mitchell, W. J., Inouye, A. S., Blumenthal, M. S., et al. (2003). *Beyond Productivity: Information Technology, Innovation, and Creativity*. Washington, DC: National Academies Press.

Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., Group, P., et al. (2009). Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. *PLoS Med.* 6:e1000097. doi: 10.1371/journal.pmed.1000097

Moreira, L. (2020). *Pixar: Where Creativity Meets, Performance Through Lean*. Available online at: https://www.pipefy.com/blog/lean-pixar-where-creativity-meets-performance/#::text=In%20order%20to%20organize%20their,idea%20until%20it's%20finally%20animated.

Morris, N. (2007). *The Cinema of Steven Spielberg: Empire of Light*. New York City, NYC: Columbia University Press. doi: 10.7312/morr476489

Netland, T. H., and Powell, D. J. (2016). *The Routledge Companion to Lean Management*. London: Taylor & Francis. doi: 10.4324/9781315686899

Nogami, T. (2006). *Waiting on the Weather: Making Movies With Akira Kurosawa*. Berkeley, CA: Stone Bridge Press, Inc.

Ohanian, T., and Phillips, N. (2013). *Digital Filmmaking: The Changing Art and Craft of Making Motion Pictures*. Boca Raton, FL: CRC Press. doi: 10.4324/9780080504407

O'Neill, E. (2001). *User-Developer Cooperation in Software Development: Building Common Ground and Usable Systems*. Berlin: Springer Science & Business Media.

Paez, A. (2017). Gray literature: an important resource in systematic reviews. *J. Evid. Based Med.* 10, 233–240. doi: 10.1111/jebm.12266

Parker, P. (1999). *The Art and Science of Screenwriting*. Bristol: Intellect Books.

Pe na-Acu na, B. (2018). *Understanding Steven Spielberg*. Cambridge: Cambridge Scholars Publishing.

Perko, M. (2018). "Origin stories: Stanley Kubrick's Collaborations," in *Essais*, (*Hors-série 4*). doi: 10.4000/essais.717

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: an update. *Inform. Softw. Technol.* 64, 1–18. doi: 10.1016/j.infsof.2015.03.007

Phillips, G. D. (2013). *Stanley Kubrick: Interviews*. Oxford, MS: University Press of Mississippi.

Piper, R. J. (2013). *How To Write a Systematic Literature Review: A Guide for Medical Students*. National AMR, Fostering Medical Research, 1–8.

Poppendieck, M., and Cusumano, M. A. (2012). Lean software development: a tutorial. *IEEE Softw*. 29, 26–32. doi: 10.1109/MS.2012.107

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach*. London: Palgrave Macmillan.

Richie, D. and Mellen, J. (1998). *The Films of Akira Kurosawa*. Berkeley, CA: Univ of California Press.

Robson, C. and McCartan, K. (2016). *Real World Research*. Social Science Research Group.

Sacks, O. (1992). Tourette's syndrome and creativity. *BMJ* 305:1515. doi: 10.1136/bmj.305.6868.1515

Saeki, M. (1995). "Communication, collaboration and cooperation in software development-how should we support group work in software development?," in *Proceedings 1995 Asia Pacific Software Engineering Conference*, 12–20.

Schmucker, C., Bluemle, A., Briel, M., Portalupi, S., Lang, B., Motschall, E., et al. (2013). A protocol for a systematic review on the impact of unpublished studies and studies published in the gray literature in meta-analyses. *Syst. Rev*. 2, 1–7. doi: 10.1186/2046-4053-2-24

Sedelow, S. Y. (1970). The computer in the humanities and fine arts. *ACM Comput. Surveys*, 2, 89–110. doi: 10.1145/356566.356568

Seffah, A., Gulliksen, J., and Desmarais, M. C. (2005). "An introduction to human-centered software engineering," in *Human-Centered Software Engineering-Integrating Usability in the Software Development Lifecycle* eds A. Seffah, J. Gulliksen, M. Desmarais (Berlin: Springer), 3–14. doi: 10.1007/1-4020-4113-6_1

Shah, R., and Ward, P. T. (2003). Lean manufacturing: context, practice bundles, and performance. *J. Oper. Manage*. 21, 129–149. doi: 10.1016/S0272-6963(02)00108-0

Siddaway, A. (2014). What is a systematic literature review and how do i do one. *Univ. Stirl*. 1, 1–13.

Slater, L. (2005). Leadership for collaboration: an affective process. *Int. J. Leadersh. Educ*. 8, 321–333. doi: 10.1080/13603120500088745

Stiglegger, M. (2001). *Donald Richie: The Films of Akira Kurosawa, 3rd Edn, Expanded and Updated With a New Epiloque*. Redditch: Westland.

Trifonova, A., Jaccheri, L., and Bergaust, K. (2008). Software engineering issues in interactive installation art. International *J. Arts Technol*. 1, 43–65. doi: 10.1504/IJART.2008.019882

Van der Lelie, C. (2006). The value of storyboards in the product design process. *Pers. Ubiquit. Comput*. 10, 159–162. doi: 10.1007/s00779-005-0026-7

Wallace (1999). *Is Software Art or Engineering*? Available online at: https://www.spectacle.org/1199/software.html

Whitehead, J. (2007). "Collaboration in software engineering: a roadmap," in *Future of Software Engineering (FOSE'07)*, 214–225. Washington, DC: IEEE. doi: 10.1109/FOSE.2007.4

Winkler, D., Mordinyi, R., and Biffl, S. (2013). "Research prototypes versus products: lessons learned from software development processes in research projects," in *European Conference on Software Process Improvement* (Berlin Springer), 48–59. doi: 10.1007/978-3-642-39179-8_5

Yakimchuk, N. (2017). *Small Budget? Producer Jason Blum's 5 Rules for Lean Filmmaking*.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.