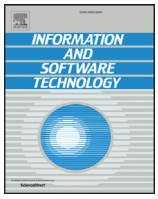
Information and Software Technology 121 (2020) 106257

Contents lists available at ScienceDirect

Information and Software Technology

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



Time pressure in software engineering: A systematic review

Miikka Kuutila*, Mika Mäntylä, Umar Farooq, Maëlick Claes

University of Oulu, M3S, Pentti Kaiteran katu 1, Oulu 90014, Finland

A B S T R A C T

Context: Large project overruns and overtime work have been reported in the software industry, resulting in additional expense for companies and personal issues for developers. Experiments and case studies have investigated the relationship between time pressure and software quality and productivity.

Objective: The present work aims to provide an overview of studies related to time pressure in software engineering; specifically, existing definitions, possible causes, and metrics relevant to time pressure were collected, and a mapping of the studies to software processes and approaches was performed. Moreover, we synthesize results of existing quantitative studies on the effects of time pressure on software development, and offer practical takeaways for practitioners and researchers, based on empirical evidence.

Method: Our search strategy examined 5414 sources, found through repository searches and snowballing. Applying inclusion and exclusion criteria resulted in the selection of 102 papers, which made relevant contributions related to time pressure in software engineering.

Results: The majority of high quality studies report increased productivity and decreased quality under time pressure. The most frequent categories of studies focus on quality assurance, cost estimation, and process simulation. It appears that time pressure is usually caused by errors in cost estimation. The effect of time pressure is most often identified during software quality assurance.

Conclusions: The majority of empirical studies report increased productivity under time pressure, while the most cost estimation and process simulation models assume that compressing the schedule increases the total needed hours. We also find evidence of the mediating effect of knowledge on the effects of time pressure, and that tight deadlines impact tasks with an algorithmic nature more severely. Future research should better contextualize quantitative studies to account for the existing conflicting results and to provide an understanding of situations when time pressure is either beneficial or harmful.

1. Introduction

Interest in scheduling and time-related issues in software engineering has been expressed for decades. In the 1970s, in the widely influential *The Mythical Man-Month: Essays on Software Engineering*, Frederick Brooks coined the idea known as Brooks' law: "adding manpower to a late software project makes it later" [20]. Similarly, textbooks from the '80s and '90s for software developers and managers, have dedicated chapters and subchapters for "deadline pressure" and "beating schedule pressure" [41,85]. More recently, it has been shown that 60-80% of software projects are late (encounter overruns) [87]; because being late is an antecedent of time pressure, we can assume the latter is fairly common in the software industry.

In psychological literature, time pressure refers to situations where time is a limited resource [84]. There are several well-validated theories related to time pressure and its effects on stress, decision making, and motivation, such as the Yerkes-Dodson law [145], the job demands-resources model [10], and the speed-accuracy trade-off [46]. These theories, which we discuss in more detail in Section 2, are relevant to time pressure in software engineering, despite having been developed in other fields.

Project overruns and overtime work in the software engineering industry are reported as common by both academic [87] and practitioner [40,121] sources. Yet, we argue that a systematic review of the current understanding of time pressure and its effects on productivity in software engineering is needed, in order to provide a more solid basis for future research. For example, by compiling together previously used metrics of time pressure, our work enables future studies to conduct well-informed measurements of time pressure in software engineering.

This review heavily extends our previous work [63], in which clustering on Scopus data was performed to identify in literature more specific topics on time pressure in software engineering and related disciplines. We partly use that work to establish a "seed" set of papers, which we complement with novel, relevant articles, with Google Scholar searches. The papers are expanded using Wohlin's snowballing guidelines [142]. Our goal is to provide an overview of the existing literature related to time pressure in software engineering and map it to different process phases, as well as to synthesize the gathered information in a way that provides new information. The latter is accomplished by seeking answers to the following research questions:

RQ1-Definitions What definitions of time pressure are used?
RQ2-Metrics What metrics are used to measure time pressure?

* Corresponding author.
E-mail address: miikka.kuutila@oulu.fi (M. Kuutila).

https://doi.org/10.1016/j.infsof.2020.106257
Received 12 July 2019; Received in revised form 9 January 2020; Accepted 12 January 2020
Available online 18 January 2020
0950-5849/

RQ3-Process Phases What process phases or approaches are studied with respect to time pressure?
RQ4-Causes What causes of time pressure are reported?
RQ5-Effects and Outcomes What are the effects of time pressure on software development?

The goals of a systematic mapping study are to provide an overview of a research topic, identify relevant quantities and type of research [105], while the goal of a systematic literature review aims to summarize and examine to what extent does empirical evidence support or contradict the considered hypotheses [60]. Hence, the first four research questions are more related to systematic mapping studies, while *RQ5-Effects and Outcomes* is applicable to systematic literature reviews. Although our paper combines elements of both, we title our work a systematic review, for convenience. The remainder of this article is structured as follows. In Section 2, we outline background information on theories and concepts related to time pressure. In Section 3, we introduce the methodology used in this study. In Section 4, we present the definitions of time pressure, discuss the different metrics that have been used, and elaborate on previous theoretical work. In Sections 4.1 and 4.2, research questions *RQ1-Definitions* and *RQ2-Metrics* are answered, respectively. In Section 5.2, we map a number of different studies to processes and approaches, to answer *RQ3-Process Phases*. Section 6 contains an overview of the existing literature and summarizes the empirical descriptives provided by several studies. In Sections 6.1 and 6.4.5, research questions *RQ4-Causes* and *RQ5-Effects and Outcomes* are answered, respectively. Last, in Section 8, we conclude the paper by outlining our contributions and providing a series of takeaways for practitioners and researchers.

2. Related work

In occupational and social psychology, time is considered to be a resource. The scarcity of resources, such as time and money, and the effects of scarcity on mindset and human perception have been a subject of popular science [89]. Mullainathan and Sharif [89] make the case that scarcity of time introduces both *tunneling* and *focus* mindsets. Tunneling, in this context, means valuing short-term over long-term goals that are related to the scarcity of a resource. An example could be when time-scarce software engineers prefer a solution that is quick to implement, regardless of its impact on the longevity of the software. However, a scarcity mindset can also make individuals focus when spending a limited resource. In our example, scarcity of time would make software engineers avoid gold plating, i.e. working on a task beyond what is reasonably expected. In many studies, time pressure is defined as the perception that time is *scarce* in relation to the demands of the task [13,28,59].

Another view on time pressure comes from the Yerkes-Dodson law, which dates back to the start of the 20th century [145], and states that arousal, caused by (time) pressure, and performance have an inverted *U*-shaped relationship. This is depicted in Fig. 1. In other words, arousal increases performance, but only up to a certain point, from where performance starts decreasing. In this view, time pressure is seen to increase the activation level and urgency (arousal).

The speed-accuracy trade-off [46] offers another lens for viewing time pressure: it is observed that the decision speed negatively correlates with the quality of the decision. In fact, the phenomenon is ubiquitously observed, with species such as ants or bumblebees [46]. The speed-accuracy trade-off is commonly used as a benchmark for decision processes across task-domains, as simply examining either reaction speed or quality alone is not a sufficient benchmark. The speed-accuracy trade-off has also been, in part, the subject of human computer-interaction studies [73].

In occupational psychology, the well-known job demands-resources model [10] is used to explain employee well-being. Generally, the model assumes every job to have both demands and resources, and well-being to be the result of their balance. Resources in the model refer to skills, autonomy, feedback, and others, while demands can include role ambiguity, performance, and emotional demands. Hence, time pressure can be seen as a demand, with too much of it leading to worse well-being outcomes, exhaustion, and even burnout [31].

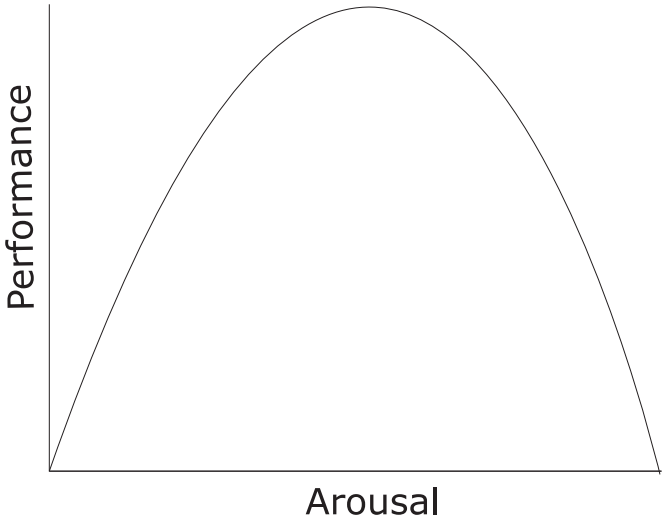


Fig. 1. The Yerkes-Dodson law.

Evidence has also shown teams may work exceptionally well under extreme time pressure, such as the Apollo 13 ground crew [25]. Thus, in a work by Chong et al. [25], a challenge-hindrance framework [69] for time pressure was deemed appropriate. In this view, time pressure is defined as having either positive (challenge) or negative (hindrance) effects on goal achievement. LePine et al. [69] pointed out that challenges could be viewed as good stress, while hindrances as bad stress. Thus, not only the amount, but also the type of time pressure matters. Examples from Chong et al. [25] of hindrance (bad) time pressure are "amount of constant switching between tasks" or "impossibility to fulfill the project schedule," while challenge (good) time pressure item examples are "importance of completing this project on time" or "urgent need for successful completion of the work the team is doing." As recognized by Chong et al. [25], the boundaries between challenge and hindrance are not always clear, and Chong et al. [25]'s survey had several items that did not clearly fall in either category. In software engineering, the challenge-hindrance time pressure definition has been used by Lohan et al. [72].

As demonstrated by the clustering performed in our previous work [63], time pressure has been studied across different occupations and fields, e.g., accounting, medicine, construction work, and vessel operating pilots. The mediating role of knowledge under time pressure haven been previously suggested: for example, accounting professionals (considered a high-knowledge group) improved their performance under time pressure, whereas the performance of students (low-knowledge group) decreased [127]. Similarly, industry-specific auditors have been reported to experience less time pressure than non-specialized auditors [49]. In nursing, time pressure has been linked to worse patient safety, but only together with high levels of burnout [132]. Another context where time pressure has been studied is related to risky choice behaviour, the majority of the studies indicating higher risk taking under time pressure [74].

3. Methodology

In this section, we present the methodology used in this study. Section 3.1 introduces the inclusion and exclusion criteria we used to grade the found literature. In Section 3.2, we present the database search strings and approaches. In Section 3.3, we explain the snowballing pro-

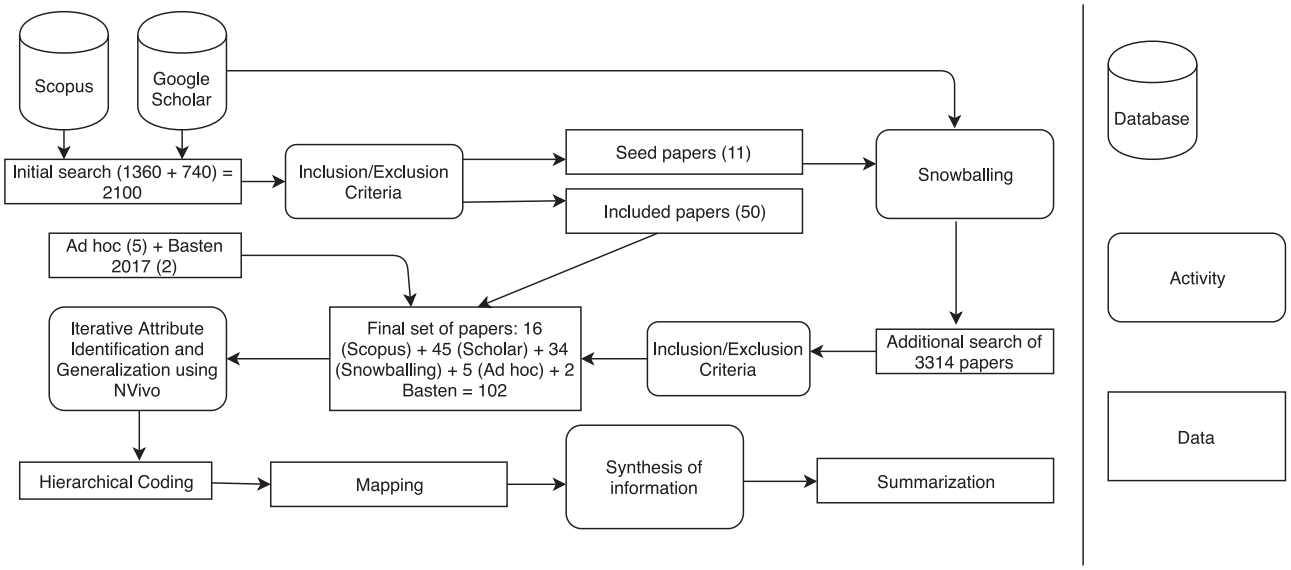


Fig. 2. Flow chart of our research methodology.

cedure in detail. Last, in Section 3.4, we explain the data extraction in detail. The entire research methodology is shown in Fig. 2.

3.1. Selection criteria and selection process

As previously stated, our goal is to provide an overview of time pressure in the software engineering context, that overviews the definitions, metrics, process phases, causes and effects found in the literature. This goal leads to our first inclusion rule (I1).

The second inclusion rule (I2) is about studies that include time pressure variables and provide empirical evidence, but whose main focus might lie elsewhere. For example, integration failures [24]. The second inclusion rule gives this paper a broad scope and highlights less well-known papers related to time pressure in software engineering that are not as easily found. In practice, we excluded papers with brief mentions of deadlines or time pressure, which were not derived based on a systematic effort to gather information, as anecdotal level evidence. For example, sources where time pressure is used to explain unexpected results ex post facto as a possible contributing factor. We define empirical evidence as either quantitative studies conducted with data including variables related time pressure, or qualitative studies which study and have practical takeaways in time pressure and time usage related issues. Hence our inclusion criteria takes into account quality of evidence.

We used the following inclusion criteria:

I1 The main focus is time pressure in software engineering.
I2 The paper presents empirical evidence of time pressure in software engineering.

We formed the first and second exclusion rules (E1 and E2) at the beginning of the search to quickly exclude sources that we could not reliably interpret, or which were not published in scientific venues. The exclusion rules E3 was formed iteratively throughout the literature search process when we encountered studies that were clearly related to time pressure and computers, but that did not relate directly to software development, such as studies on the end users of information systems. We used the following exclusion criteria:

E1 The paper was written in language other than English.
E2 Not a scientific source.
E3 The task studied is not a software development task.

Only one of the inclusion criteria had to be present in a paper for it to be included. In practice, this meant that papers without empirical evidence had less chance to be included, as their focus had to be time pressure to be included with inclusion rule I1. One such example is the agency model by Austin [7], which only satisfies the inclusion rule I1. However, multiple papers we included fulfill both rules of the inclusion criterion.

The use of exclusion criteria differed based on the rule. Rules E1 and E2 excluded a paper as soon as we observed the paper was written in a language other than English or that it was not published in a scientific venue. However, when we discovered papers with elements of exclusion criterion E3, we examined them so that we could come to a verdict. In other words, if the papers made contributions related to I2, but all these contributions were covered by exclusion criteria E3, the paper was excluded. However, if the paper made some contributions related to inclusion criteria I2, for which exclusion criterion E3 was not related, we included the paper.

The selection process for each paper started with the first author reading the title and the abstract of the paper. In cases where no definitive decision could be made based on these elements, the first author read the conclusions and other relevant sections of the paper until we could make a decision based on the selection criteria. The first author was advised to default to caution when unclear cases were found, as we did not want to miss any potential papers. This made the selection process more laborious, but reading just the abstract would not have uncovered papers making significant and relevant contributions such as Cataldo and Herbsleb [24].

Unclear or borderline cases were marked down and discussed by the first two authors until a decision for inclusion or exclusion could be made. We used two person review only for borderline papers. We settled upon this approach as we had very large set of papers to cover and as most of the exclusion decisions were simple, as can be seen in Table 2. This meant that we could afford careful pondering on the borderline cases. For transparency, also all borderline cases are marked in our *replication package*.[1] A total of 129 papers were marked unclear and discussed by the two first authors. Of these 129 papers, we included 23 in the literature review. Occasionally, the unclear cases led to clarification of the interpretation of the inclusion and exclusion rules. For example, we did not anticipate that there would be papers from non-software engineering forums that gave in-depth narrative descriptions of software companies with extensive time pressure and its consequences,

[1] https://figshare.com/s/0662c66e0705ebf8dca7.

Table 1
Search strings, total results and included papers from the database search.

Search Engine	Search String	Results	Included
Scopus	Title or keyword: "time pressure", "schedule pressure" "time budget pressure", "deadline pressure" "pressure of time", "pressure of schedule" "pressure of time budget", "pressure of deadline" "speed accuracy tradeoff" And not title, abstract, or keyword: "long rise-time", "intracranial", "drill", "space-time"	1270	12
Scopus	"schedule compression" AND "software engineering"	91	4
Google Scholar	"time pressure" AND "software engineering"	100	13
Google Scholar	"schedule pressure" AND "software engineering"	100	16
Google Scholar	"time budget pressure" AND "software engineering"	100	3
Google Scholar	"deadline pressure" AND "software engineering"	100	11
Google Scholar	"pressure of time" AND "software engineering"	100	0
Google Scholar	"pressure of schedule" AND "software engineering"	19	0
Google Scholar	"pressure of time budget" AND "software engineering"	4	0
Google Scholar	"pressure of deadline" AND "software engineering"	17	1
Google Scholar	"speed-accuracy tradeoff" AND "software engineering"	100	0
Google Scholar	"schedule compression" AND "software engineering"	100	1

Table 2
Verdicts.

Verdict	Scopus	Scholar	Snowballing	%
The main focus is time pressure in software engineering (I1)	10	5	11	0.5%
The paper presents empirical evidence of time pressure in software engineering (I2)	6	40	23	1.3%
Already Included, duplicate	4	35	81	2.2%
No empirical evidence on time pressure, not focused on time pressure (I1 & I2)	73	523	1366	36.2%
Not a software development paper (E3)	1246	109	1491	52.6%
Not a scientific source (E2)	15	10	19	0.8%
Not available	7	17	194	4%
In language other than English (E1)		1	128	2.4%

such as Borg [19], who observed an ICT company and the problems arising from time pressure. The first author went through all the sources found with database searches twice. This was to ensure our interpretation of the rules were consistently enforced, additionally we had not explicitly recorded the reasons for inclusion and exclusion during the first round.

3.2. Database search methodology

Part of the initial database search was based on our previous work [63], in which we used the Scopus database search with Latent Dirichlet Allocation (LDA) clustering to get an overview of where time pressure has been studied. As advocated by Kitchenham and Charters [60], we used multiple trial searches to establish keywords and synonyms for time pressure, all of which can be found in Table 1. We used Scopus and Google Scholar for the primary literature search from databases. All used search strings can be found in Table 1, together with the number of total results and the number of primary sources we included for our review. Because Google Scholar also searches in the article full-text, while Scopus searches only for title, abstract, and keywords, we used the tool *Publish or perish 5*[2] to complement the search from Scopus with Google Scholar searches. For the complementary Google Scholar search, we took only the first 100 results for each search. Additionally, we performed some ad hoc searches, and they resulted in six additional papers. Altogether, we included 61 sources from the 2100 found with repository searches. In the flowchart described in Fig. 2, the initial search can be found in the activity "Initial search," and the additional papers found with ad hoc searches are added to the final set of papers as "+7."

[2] https://harzing.com/resources/publish-or-perish.

3.3. Snowballing

Wohlin's snowballing guidelines [142] were used to construct a set of seed papers and snowball through them. The set of papers used for snowballing contained papers we found with repository searches (Section 3.2). We included 11 papers, with broad range of authors and publication fields and venues. One round of snowballing was performed forward and backward for these papers. Additionally, all publications published by the authors included in the seed paper set were examined with the selection criteria. We used only Google Scholar to identify papers needed for the snowballing. Indeed, we had determined that effort was better spent on reviewing more papers on the snowballing phase rather than double checking two search engines (Scopus and Google Scholar) for a smaller number of additional papers.

In the snowballing phase, 3314 papers were evaluated based on the selection criteria, and it resulted in the inclusion of 34 additional papers. The snowballing phase had an inclusion rate of 1.03%.

Before we started hierarchical coding, two of the authors went through all the included papers together and discussed their relevance for the topic. Based on these discussions we excluded six papers, as in practice these papers included only anecdotal evidence. These six exclusions are not reported in the previous numbers.

While we were conducting this literature review, work by Basten [13] has been published. Our work at that stage already included 9 of the 13 papers Basten introduced in his work. Of the four not included, based on our inclusion and exclusion criteria, we decided to include two, as well as the paper by Basten. This explains the +2 in Fig. 2 in the final set of papers. Overall, our work is broader than Basten's as we have 102 papers while Basten had 13 of which we include all but two.

We have also provided verdicts for all papers in Table 2. This table only includes verdicts for papers found with the database search strategy and snowballing. This means that papers found Ad hoc (see Fig. 2), such as Basten [13], are not included in the table.

As can be seen in Table 2, the most common reason to exclude a paper was when its context was deemed outside of software engineering. This, for example, included papers focusing on users of information systems rather than developers. The second most common reason to exclude a paper was not meeting any of inclusion criteria. Not available verdict includes sources with incomplete bibliographic information, non-digitalized material, as well as some papers behind paywalls we did not have access to. Of the included papers, 29 were included with the inclusion criteria I1 and 73 were included with the criteria I2.

3.4. Data extraction

The papers we included were qualitatively coded and analyzed using QSR International's NVivo.[3] We followed Zhang and Wildemuth [147] qualitative guidelines. The coding scheme was both used to familiarize ourselves with the material, as well as to form the basis for Sections 4, 5.2 and 6.

Coding started with few ready codes, such as process phases and causes of time pressure, but it was iteratively improved. In the end, highest level of the coding included classes empirical results, definitions, research methodology, research questions and hypotheses, and lastly process phases. The coding for process phases formed the basis for the mapping in Section 5.2, while coding for empirical results formed the basis for Section 6. In the first round, the first author coded all the sources. The resulting coding scheme was improved and checked by the second author. Once every paper was coded and the coding scheme was ready, the first author went through all the papers a second time to apply the scheme consistently on every source. Finally, all the authors checked the results and analyzed a part of the coding that was given to them. In practice, this meant that at least two persons checked all parts of the coding scheme and coded text. We have included a picture depicting the coding scheme in our *replication package*.[4]

4. Definition, metrics, and previous theoretical work in software engineering

In this section, we provide and examine the definitions given for time pressure in the previous literature, summarize the metrics used to measure it, and introduce prior theoretical works focusing on time pressure in software engineering context. Research question *RQ1-Definitions* about which definitions of time pressure are used is answered in Section 4.1. Additionally, research question *RQ2-Metrics* about which metrics are used to measure time pressure is answered in Section 4.2.

4.1. Definition

Multiple synonyms for time pressure exist in the scientific literature about software engineering, such as schedule pressure [34], deadline pressure [29], and time budget pressure [91]. Schedule pressure and deadline pressure emphasize a deadline or deadlines when a task or project should be done while time budget pressure highlights the amount of time that can be used for a task or a project.

Pressure due to compression of the schedule was considered in the early work of software engineering. Barry Boehm [18] defined schedule compression as the percentage of schedule cut in a project's planned duration compared to the nominal schedule of the project. This a definition was used later in research into project simulation models [1,53].

Powell et al. [108] defined schedule pressure as the relationship between required and applied productivity. Additionally, they discovered that "it is possible to increase pressure on the development team (by requiring additional productivity) but only up to a certain point, after which productivity rapidly declines," pointing at the Yerkes-Dodson law [145].

To summarize, the definitions of time pressure focus on an individual's perception of the time scarcity [13,28,59], or on the project level of schedule compression [18,108]. The two main concepts of how time pressure is understood in software engineering literature are the *U*-shaped relationship between arousal and performance [145] and the division of time pressure into challenge and hindrance time pressure [25].

4.2. Metrics and operationalization

Gathering metrics for time pressure is essential. As different definitions of time pressure exist, metrics for operationalization of those definitions reveal more details about time pressure. The metrics in empirical settings can vary from study to study, and they also depend on what is available in each context. Summarizing the metrics helps future researchers as they can consider the existing metrics when designing their studies. Table 3 shows all the metrics we found to measure time pressure.

Questionnaire and survey-based metrics that use an ordinal scale to measure time pressure are popular [97]. Early on, Banker and Kemerer [11] simply asked project leaders if the deadline pressure was higher than average. Similarly, Mukhopadhyay and Kekre [88] asked to rate software projects on a scale of one to four regarding deadline pressure, where four signaled very high pressure. Durham et al. [32] developed a scale specifically designed to measure time pressure, which Maruping et al. [83] used to study time pressure in software projects. General questionnaires, such as the NASA Task Load Index (NASA-TLX), have a question about the temporal demand of a task and have been used to measure time pressure in software engineering [81].

The literature of software cost estimation uses metrics-based estimated effort. Ruiz et al. [117] defined schedule pressure as estimated effort, minus the remaining effort divided by the estimated effort, meaning a positive value indicates delayed project, while a negative value indicates a project that is advancing according to the initial estimates. Nan et al. [90] defined time pressure as the estimated time for the project minus the customer negotiated time for the project divided by the estimated time for the project. This work is similar to Ruiz et al. [117], but the difference is that they used customer-negotiated time instead of estimated effort. Cycle time was defined as the project duration starting from the first day of design work and continuing until the customer accepts the delivered product. The authors also used this metric in a more well-known later work from 2009 [91]. In COCOMO II [17] actual schedule compression is defined as the ratio between actual schedule and the estimated nominal schedule.

Cataldo [23] used a metric, called *the task temporal metric*, to estimate time pressure experienced in a project. It is calculated as the standard deviation of modification requests completed each month. The author contented that the high value of the task temporal metric is associated with an uneven workload during the project which suggests time pressure in the months with a high number of tasks.

More recently, there have been efforts to detect time pressure with sentiment analysis and various sensors. Kołakowska et al. [62] introduced a multi-modal emotion recognition application, which combined physiological, video, and depth sensors, to train a classifier to be used with several software engineering methods. The motivation for the work in part came from future work of detecting stress induced by time pressure and an investigation of productivity and emotions. Similarly, using sentiment analysis, Mäntylä et al. [77] found that higher arousal (e.g., activation level) was associated with more severe issue reports. In a later work, Mäntylä et al. [80] developed a lexicon for sentiment analysis for more efficient detection of arousal levels in the software engineering context.

We present experimental designs used to create time pressure; see Table 4. They are not about measuring time pressure but represent essential information on the operationalization of time pressure. An early paper by Hwang [52] noted that time pressure can be operationalized as

[3] https://www.qsrinternational.com/nvivo/home.
[4] https://figshare.com/s/0662c66e0705ebf8dca7.

Table 3
Metrics identified from the collected literature.

Metric to Measure Time Pressure	Example Papers	Data Source
$\frac{\text{Estimated time - customer negotiated time}}{\text{estimated time}}$	[90,91]	Company Project Database
$\frac{\text{Actual Schedule}}{\text{Nominal Schedule}}$	[17]	Company Project Database
$\frac{\text{Estimated effort - Remaining effort}}{\text{Remaining effort}}$	[117]	Model simplification
Standard deviation of tasks completed in a project each month	[23]	Company Project Databases
Questionnaires and surveys	[83,97]	Questionnaires and surveys
Physiological measurements	[62]	Skin Conductance
	[134]	Electromyography Electrocardiography, etc.
Sentiment analysis	[77,80]	Natural Language Text

Table 4
Creating time pressure in experiments.

Creating Time Pressure	Example Papers	Data Source
Time limits in experimental settings	[56,78]	performance in the experiment
Task difficulties in experimental settings	[113]	performance in the experiment
Reward for faster completion in experimental setting	[81]	performance in the experiment

the time available for task performance, for example, as different time limits in experiment settings. Different time limits [56] and task difficulties [113] have been widely used in experimental settings as operationalizations of time pressure. Rewarding faster completion is another alternative to create time pressure in experiments [81].

4.3. Theoretical papers and reviews

During our search for academic literature, we did not find any previous sources following systematic literature review guidelines to assess previous work related to time pressure in software engineering. However, we found theoretical work and non-systematic literature reviews that focused mainly on time pressure in software engineering.

Early focus on time pressure in software engineering is related to cost models and cost estimation. Costello's paper from [29] is a prime example of a purely theoretical paper. The paper presents a simplistic scheduling model and discusses schedule pressure at length based on experiences. The main contribution of the paper is a list of resource allocation strategies aimed at decreasing the effects of schedule pressure.

Widely cited paper by Austin [7] presents an agency framework focused on the effects of time pressure on software quality. Based on the modeled framework, the author recommends setting aggressive deadlines, where it is okay to miss deadlines. The author also concluded that adding slack time does not necessarily minimize costs and that deadlines should be set separate from planning estimates.

Malgonde et al. [75] presents a research proposal about how emergent outcome controls are adapted when time pressure increases. The authors planned to investigate with interviews and critical incident method.

Harris et al. [44] considers time-related concepts and issues in Agile software development and introduces research propositions for the future. These concepts and issues are highlighted with data from qualitative interviews. The authors compare the role of deadlines in traditional software projects that use the waterfall life cycle and software projects that use Agile methods. The authors argue that employee motivation and stress should be compared with multiple shorter deadlines versus one longer deadline in the Agile software development context, and additionally comparing time-to-completion with higher and lower uncertainty projects developed with Agile and plan-driven approaches.

Basten [13] present a literature review and a research agenda basis for future studies. Part of the research agenda is a call for methodological pluralism, as the author argued that previous works did not use qualitative research methods. Basten [13]) also argue for research agenda conceptualization (e.g., better definitions of time pressure), research on contemporary development approaches, such as Agile and Scrum, better definitions of the role of the context of time pressure to better understand the diverse results, and empirical validation in the form of replication studies.

Last, our previous work [63] presented a computer-aided literature review and introduced testable hypotheses related to time pressure from fields other than software engineering. The paper presented a list of testable hypotheses in software engineering related to time pressure derived from fields other than software engineering. For example, under time pressure, fewer test can run and hence less feedback is provided, or more bugs are introduced to the code in more complex classes near deadlines. Additionally, that paper formed the foundation for this paper.

5. Mapping

In this section, we provide bibliographical information the selected studies, map them to process phases and methodologies, as well as to used research methods. In Section 5.1, we provide bibliographical information for the selected studies and their mapping to the adopted research methods. Next, Section 5.2 discusses the mapping to process phases and methodologies.

5.1. Publication years, venues and used research methods

Fig. 3 shows the publication years of all the papers included in the present study. It can be observed that the number of publications is larger after the year 2000, than before it. All the earliest publications focus on cost estimation and process simulation models, e.g., Costello [29] and Jeffery [53]. More recent papers include Agile methodologies (e.g., Laanti [65]), software process improvement (e.g., Agrawal and Chari [4]), and time pressure detection (e.g., Kołakowska et al. [62]).

Similarly, Table 5 shows the most common publication venues, which published two or more papers of the articles we included in the study. We note that 35 were published in conferences or workshops, 57 in academic journals, 7 in academic magazines (e.g., IEEE Software, Computer, Communications of the ACM, etc.) and 2 in Ph.D. theses.

We have mapped the included studies also according to the adopted research methodologies. We define as "Theoretical Papers & Reviews" (see Table 6) articles that do not provide their own original empirical evidence (i.e., secondary and tertiary studies). Next, position pieces include positional papers arguing for a position or a research proposal, but lacking experimentation and original research. We have defined studies

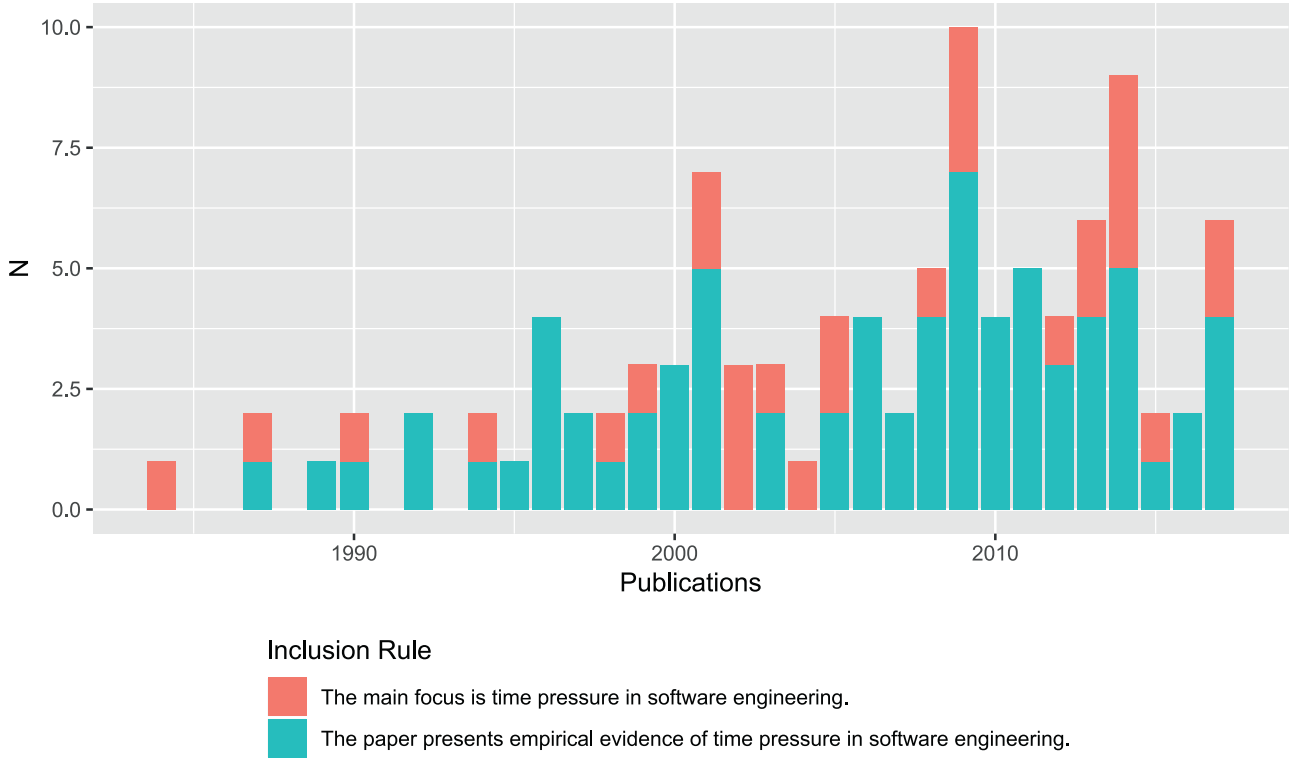


Fig. 3. Number of publications per year.

Table 5
Most common publications venues.

Publication	*N*
Journal of Systems and Software	10
Information and Software Technology	7
International Conference on Software Engineering (ICSE)	5
IEEE Transactions on Software Engineering	5
Americas Conference on Information Systems (AMCIS)	4
Academy of Management Journal	3
Computer	3
International Symposium on Empirical Software Engineering and Measurement (ESEM)	3
Journal of Management Information Systems	3
Communications of the ACM	2
CrossTalk - The Journal of Defense Software	2
Hawaii International Conference on System Sciences (HICSS)	2
IEEE Software	2
Information Systems Research	2
International Conference on Information Systems (ICIS)	2
International Conference on Mining Software Repositories (MSR)	2
International Journal of Human-Computer Interaction	2
Others	43

that create and evaluate an artifact (e.g., a simulation model) as design science research, as defined by Peffers et al. [100]. Otherwise, for empirical research we followed the categorization by Easterbrook et al. [33], with the addition of a "company data" category for studies that quantitatively analyze software project data, usually from multiple projects. We present the result in Table 6. The most commonly used research method is Design Science. The code to reproduce Tables 5 and 6 and Fig. 3 is given in our *replication package*.[5]

5.2. Process phase or approach

In this section, we summarize and map the papers we found into different process phases and methodologies, to give the context of their results. The results are shown in Table 7. This can be useful for a scientist who is interested in a particular process area. We map the papers to the different process phases of the waterfall model [126]. Additionally, we mapped the sources to two other categories. First, the category *whole process or approach* includes papers that cover multiple process phases. Second, the category *other* includes various sub-categories in which multiple papers concentrated on a single theme. One paper can be mapped to multiple groups.

We found only two papers related to requirements engineering. In total, the software design category contains five papers and includes papers related to software acquisition. Papers related to programming and implementation, in general, are grouped under one category and included five papers. Papers related to integration, testing, and defect fixing are categorized under quality assurance and included 17 papers in total. For the process phases, papers related to maintenance are cat-

[5] https://figshare.com/s/0662c66e0705ebf8dca7.

Table 6
Number of implemented research methodologies.

Methodology	N
Design Science Research	19
Mixed Methods	14
Survey	13
Experiment	12
Case Study	12
Company Data	12
Theoretical Papers & Reviews	9
Position Piece	7
Ethnographic Study	3
Action Research	1

egorized with software evolution and release engineering and include three papers. Of all the sources categorized into software process phases, the highest number of papers is found in the quality assurance category. This could reflect that quality assurance is the last step before software releases, and thus, is performed just before the most critical deadlines.

Other sub-categories are derived from the qualitative coding with NVivo. We grouped papers related to whole processes or approaches into Agile and Scrum, project failure, and process improvement. In addition, papers related to temporal aspects of project management are in this category as subcategory cost estimation, cost models, simulation, and project escalation. There is a total of 29 papers in this category. The high number of studies related to cost estimation, cost models, simulation, and project escalation can be explained by the problems in these activities, as errors in cost estimation and scheduling cause time pressure. This is further explained in Section 6.1.

Last, we put other groups under the group *Other*. *Detection of time pressure* includes papers that present or investigate ways of detecting hurry and arousal in software engineering. *Group interaction* contains sources investigating interaction in software context. Sources not containing their own empirical evidence are grouped as *Literature reviews and theoretical papers*. Papers from non-software engineering fields such as psychology, occupational health, and sociology as a fourth subcategory. We included these papers because they examine software engineering projects and offer valuable contributions to the understanding of time pressure in software engineering context. We also found three papers examining new product development and five papers that investigated group interaction in software engineering. We placed papers from studying factors related to the mind in the sub-category *Individual psychological factors*.

6. Empirical results

In this section, we present and summarize the empirical contributions of the papers we found in the literature. First, we review the identified causes of time pressure in Section 6.1. Then, we present effects on individuals in Section 6.2 and different software processes in Section 6.3. Finally, in section 6.4, we review how time pressure affects the outcome of a software project by investigating the results through common project management measures of time, cost and quality. Each of these sections end with a summary helping a skimming reader.

6.1. Causes of time pressure

6.1.1. Effort estimation, scheduling, and management

Several problems in effort estimation and scheduling that cause time pressure are mentioned in the literature. These problems include long schedules [55], insufficient experience [139], insufficient historical data [123], schedule slips [136], and change requests [54,67].

Jones [55] wrote about the reasons for software project failures. In his article, he listed the root causes of unrealistic schedule pressure as follows: “1. Large software projects usually have long schedules of more than 36 months. 2. Project managers are not able to successfully defend conservative estimates. 3. Historical data from similar projects is not available. 4. Some kind of external business deadline affects the schedule.”.

In further work, Ebert and Jones [34] noted that projects with higher defect removal effectiveness tend to have shorter schedules, as testing is the part of development where delays typically happen. This observation is supported by Table 7 that shows a high number of papers in the quality assurance phase. The authors elaborate further: “Applications that enter testing with an excessive volume of defects cannot exit the testing phase because they don’t work.”

Incorrect estimates often lead to deviations from the initial project plan [14]. Another study showed that allowing schedule slippage increases time pressure if the final deadline is not adjusted [136]. Although the root cause of schedule pressure in Van Oorschot et al. [136] was chronic under-staffing, the schedule slips increased the overall schedule pressure of this new product development project.

Reasons for effort estimation problems and subsequent failures to meet deadlines in Capability Maturity Model Integration (CMMI) level 5 organizations are given by Smite and Gencel [123]. In many cases, these problems stem from a lack of historical data for creating the estimates. Similarly, Miranda and Abran [86] suggested using probabilistic models to combat underestimation. In an older survey about using expert judgment as an estimation tool [50], it was found that only few estimators use information about deadline pressure when producing estimates.

Verner et al. [139] investigation on the causes of failures showed that software projects failed because of multiple factors. However, three of the four most common factors involved time and were outlined as the “delivery date impacted the development process,” which was present in 93% of failed projects, “project was underestimated”, which was present in 81% of failed projects, and “staff not rewarded for working long hours”, which was present in 73% of failed projects.

In a survey, partly by the same authors, time related reasons such as “project completed on time” are not seen as important for project success [110]. In contradiction, a literature review summarizing project success factors [92] lists “Realistic schedule” as the third most common factor, and “Realistic budget” as the eight most common success factor on the reviewed literature.

Change requests, especially ones that are tied to internal dependencies of the developed software, have also been reported to increase time pressure [54,67]. Similarly, requirements volatility has been mentioned as a cause for time pressure in a study conducted with surveys [36]. Overscoping, i.e. requiring more resources than available, is mentioned as a cause for time pressure by Bjarnason et al. [15]. It has been noted by Reichelt and Lyneis [114] that in complex projects with significant overruns, budgets were consumed for original work and any rework resulted in overruns and time pressure. Projects with budgets in which resources remained after initial rework were tied to less severe overruns [114].

Similarly, an unexpected shortage of resources, for example, unplanned leaves, have been reported to increase time pressures in software testing [122]. In the same paper, postponement of deadlines was also mentioned as one cause of time pressure. A case study investigating the effect of rapid releases on software testing found that testing becomes more deadline oriented with rapid releases, as testing is performed closer to deadlines [79].

In the interviews, poor planning and a lack of organization were also mentioned as a cause of time pressure in addition to redundant meetings taking time away from other work tasks [30]. In interviews conducted by Blackburn and Scudder [16], managers mentioned that by not giving developers enough time, they forced developers to reuse more code. Last, interruptions and the increased cognitive load of task switching during a constantly changing software project was mentioned by Sawyer and Southwick [120] as a reason for increased pressure and declining performance. Scheduling does not generally take individuals and their

Table 7
Found papers by investigated software development process phases and methodologies.

Category	Sub-Category	Papers
Process phase	Requirements engineering	[15,36]
Process phase	Design and Acquisition	[35,52,95,116,130]
Process phase	Programming and implementation	[82,107,119,125,133]
Process phase	Quality Assurance	[12,23,24,26,30,34,43,56,58,67,70,77–79,81,122]
Whole Process or approach	Evolution and maintenance:	[11,43,113]
Whole Process or approach	Agile and Scrum:	[15,44,65,72,76,115,116,134,138]
Whole process or approach	Process improvement	[4,8,9,93,94,99,123]
Whole process or approach	Cost estimation, cost models, simulation and project escalation:	[1–3,14,15,22,29,47,48,50,51,53,61,68,71,86,88,90,91,106,108,112,114,116,117,124,143,144,146]
Whole Process or Approach	Project Success and Failure	[34,55,92,110,139]
Other	Detection of time pressure	[62,77,80]
Other	Group interaction	[66,72,82,83,97]
Other	Fields other than Software Engineering and Information Systems	[19,27,38,39,54,101–104,120,128,131,141]
Other	Literature reviews and theoretical papers:	[7,13,29,44,63,75]
Other	New product development	[16,136,137]
Other	Individual psychological factors	[38,39,42,65,98,134,135]

long-term tasks into account, resulting in difficulties prioritizing and multitasking when the task is continuously changing.

To **summarize**, evidence shows that poor effort estimates lead to time pressure in software engineering. Conversely, realistic schedule is seen as a critical success factor for projects. A lack of historical data causes poor estimates, as well as business motivations for earlier deadlines, but more in-depth analysis for the reasons of poor estimates is beyond the scope of this work. If the final deadline is not adjusted, changes in a project's internal schedule do not help in dealing with time pressure. Moving deadlines up also increases time pressure. Based on Ebert and Jones [34] and the numerous studies in Table 7, it appears that time pressure is most common in quality assurance. The lack of a buffer for unexpected work (e.g. change requests) or unexpected resourcing changes (e.g. unplanned leaves) leads to time pressure. Finally, poor organization of work and interruptions cause time pressure in software projects.

6.1.2. Company culture

Many papers reported time pressure and long hours as part of company culture, instead of time pressure being created due to shortcomings in effort estimation, which can be technical. Prolonged or constant pressure can lead to an unsustainable pace of software development and crisis mentality in the company.

Perlow [101] reported on the use of demands by senior managers to junior managers and engineers, where those who demonstrate prioritization of work over their lives outside work are rewarded. This is also demonstrated by pressuring employees not to take time off during "crisis time," which leads to canceled vacations.

A company culture of individual heroics, high-visibility work and valuing of commitment to the company over everything were seen as factors that affect company culture and time pressure in another study by Perlow [102]. Individual heroics refer to a cycle observed in the company where deadlines are confronted too late and met with a crisis mentality and the extra effort of individual workers. High-visibility work was seen as a way to advance in the company; managers prioritized this work as they considered it crucial for their advancement, and resulted in regular checkups by the managers.

Perlow [103] also studied the time usage of software engineers in three different sites (China, India, and Hungary), to see if different cultural settings and management styles influence work hours or if long work hours are inherent in software engineering work. Perlow discovered variations in the way work is scheduled and coordinated, as well as in the flexibility of when and where software engineers can work. Furthermore, Perlow found that specialized roles and personal modes of coordination make working hours more strict, as developers need to work more overlapping hours.

Perlow et al. [104] conducted an ethnographic study of an Internet start-up during a period of 19 months, during which the company grew from a group of four students to a $125 million company to bankruptcy. Because of the context of Internet start-ups, the company adopted a culture of fast decision making. Initially, it helped the company grow, but eventually, this mentality trapped the company in a process where they believed they had to make continually faster decisions to survive. The managers decided to "light a fire under the company" to create a "state of emergency address" to stimulate people. It created a sense of urgency which had a positive influence on the speed of decisions. Faster decisions created faster growth, which itself implied a need for faster decisions. This is the opposite of intuition and theory as, after initial growth, there should have been a lower sense of urgency according to previous work [21]. The authors justified this difference with the context in which the company evolved: the fast world of the Internet.

Tapia [131] conducted a study using qualitative social research methods on the role of myths in the IT workplace. Tapia noted that in the company, employees working in teams challenged each other and developed a "one-up-man-ship" culture, where employees competed to see who could spend the most time for work. This turned into more frequent 80-hour workweeks, as free time was expected after the next deadline but most of the time was not realized.

In a study by Jemielniak [54], a similar company culture was reported, where managers believed programmers showed their commitment to the company by remaining at the workplace for longer hours. Additionally, in part because programmers' schedules were flexible at the company, and because programmers were asked to estimate how time-consuming their tasks were going to be, some software engineers admitted that scheduling and estimating changed to guessing the wishes of managers and the customer.

An action research study by Borg [19] focused on company culture in an ICT company located in Malta. Borg noted that using long work hours as a benchmark for ideal workers led to time pressure and even to burnout. Borg also noted the unequal effects on different kinds of workers, with mothers having the most trouble committing to the extra time demanded.

In **summary**, qualitative studies report on company cultures which foster time pressure. Factors reported to create company culture of constant time pressure include demanding prioritization of work over private lives to advance ones career [19,54,101], focus on individual

heroics over development process [102], specialized roles and personal modes of coordination [103], prioritizing a sense of urgency without a period of rest and refocus [104].

6.2. Effects on individuals

Positive effects of time pressure can include an increase in motivation or teamwork. Paul and He [98] report, based on an experiment, that in the context of short-term online virtual groups, time pressure enhances motivation. Similarly, Marques et al. [82] reported that pressure acts as a support that triggers teamwork.

Two studies [81,113] reported support for a mediating role of knowledge to perceived time pressure by an individual, that is, knowledge increases efficiency and decreases the effort needed for task completion.

Langer et al. [66] and Maruping et al. [83] studied the effect of managers on team performance. Langer et al. [66] studied the relation of managers' practical intelligence and job performance, finding that software projects with schedule pressure benefited from a manager who scored high on practical intelligence. Practical intelligence is related to "resolving unexpected and difficult situations that often cannot be resolved using established processes and frameworks." Similarly, Maruping et al. [83] showed that "managers can intervene to reorient team members' efforts toward effective task management through scheduling of interim milestones, synchronization of tasks, and restructuring of priorities,", increasing team performance.

In the job demands-resources model, stress is assumed to be the result of an imbalance between demands and resources, for example, the demand for the tasks needed for the next deadline and the limited time resources before it. Hence many effects of stress can be linked to a lack of time. Sanjram and Gupta [119] showed that programmers with time constraints experienced more significant workload as measured by the NASA-TLX assessment tool [45]. Furthermore, the group with time constraints failed more often when working on a separate task simulating multitasking. Fehrenbacher and Smith [35] found that time pressure increases feelings of uneasiness and willingness to postpone decisions and decreases individuals' confidence.

Taking shortcuts has been linked to time pressure in software engineering. Sojer et al. [125] reported that perceived severity of time pressure affects individuals' attitude toward unethical reuse of code, meaning developers who feel severe time pressure are more likely to have a more positive view on copying code unethically from the Internet. Similarly, on a survey by Turley and Bieman [135], shortcut taking was identified as the second most negative software development competence.

Agile ways of working have been reported to mediate the effects of time pressure. Mann and Maurer [76] reported decrease of overtime work and an increase in customer satisfaction with the introduction of Scrum. Laanti [65] reported that sustainable pace in Agile and Kanban projects lead to better performance, and that employees who reported being empowered were able to deal better with stress. Tuomivaara et al. [134] found that developers who followed agile development process more closely felt less job strain at the end of the study period.

Time pressure was mentioned as the second most frequent reason for unhappiness among software developers in a survey conducted by Graziotin et al. [42]. Additionally, time pressure was the most frequent cause of unhappiness from factors external to the developer. The most common reason was "being stuck in problem-solving".

Fujigaki [38] investigated software engineers and their well-being by using the self-reported depressive scale (SDS) and semi-structured interviews to gauge job and life events. The self-reported depressive scale was developed by Zung in [148] and has been widely used. Fujigaki [38] observed that SDS scores rose with increased job events, one of which was time pressure caused by deadlines. This result links time pressure to depressive symptoms, while higher depressive symptoms in turn have been linked to clinical depression. In a later study, Fujigaki and Mori [39] investigated physiological metrics in relation to the work strain of information system engineers. In the study adrenaline increased before a deadline, at the start of a project, and during budget negotiations. Cortisol, which captures exhaustion, increased after constant busy states, such as after deadlines and/or after employees had gotten used to the job.

Borg [19] observed that time pressure and increased working hours lead to burnout for individuals. Borg also noted the difficulty in balancing work and life outside of work and different attitudes between genders. Young mother reported being tired all the time, when her tasks continued at home with cleaning, preparing meals and taking care of the children. In contrast, men in the company mentioned their children in relation to leisure time.

Mäntylä et al. [77] detected higher arousal (i.e., activity level) with sentiment analysis on more severe issues reports in JIRA repositories. Additionally, as issues are resolved, arousal drops, offering possible ways to detect time pressure

In **summary**, it has been shown that while time pressure can have positive effects on software developers, like increased motivation, it has also negative effects. These, such as increased stress and unhappiness, can eventually lead to depression and burnout. However, these negative effects on individuals can be mediated in different ways. In particular, Agile methods such as Kanban and Scrum decrease overtime and allows developers to better deal with stress and job strain. In addition, individuals' knowledge and managers' skills can also ease the negative effects of time pressure on individuals.

6.3. Effect on the software process

Speed accuracy trade-off [10] and the covariance of decision speed with decision accuracy [46] can be seen as the general theory related to time pressure and software engineering. However, conflicting with general theory, the reported effects of time pressure on decision quality in software engineering literature are mixed. Less organizational change, communication, and knowledge transfer have been reported with time pressure. Additionally, time pressure has been reported to be an obstacle for software process improvement (SPI) and user involvement in the design process.

6.3.1. Effect on software design

Rosenberg et al. [116] discuss schedule compression in resilient agile context at length, and consider the use of parallel development as the the most powerful practice. In the authors view, the amount of schedule which can be compressed is a trade-off between the amount of feedback and planning. The lowest cost is somewhere in the middle, i.e. having both some planning and feedback from the customer.

In a survey by Tang et al. [130], the respondents considered the lack of time or budget to be the most common reason for not documenting design rationale. Similarly, Rahmandad and Weiss [112] report that under time pressure people tend to take shortcuts on documentation and requirements development. In a survey of the Chinese software industry, by Yang et al. [143] showed that one barrier to using software estimation cost models is schedule pressure. In the survey, the respondents proposed this answer; it was not a predefined answer option.

6.3.2. Effect on communication and coordination

Time pressure also affects the communication and coordination within or outside the organization. Scientific evidence from software engineering suggests that with time pressure, there is more willingness to report bad news, less knowledge transfer, less communication between testers, and less organizational change.

Based on an experiment Park et al. [97] reported that individuals under time pressure were more willing to report bad news, as well as more likely to perceive themselves having personal responsibility to do so, which were in line with previous studies on the subject Waller et al. [140]. However, less communication between developers and tester have been reported under time pressure [122]. Karhu et al. [58] found

Table 8
Empirically derived assumptions of cost and simulation models on schedule compression.

	Compressed schedule		Eased schedule	
Model	Effort	Quality	Effort	Quality
Putnam model and SLIM [96,111]	Increase		Decrease	
COCOMO I [18]	Increase		Increase	
PRICE-S PRICE Systems [109]	Increase		Increase	
SEER-SEM Fischman et al. [37]	Increase		Increase	
System Dynamics Model [1]	Increase	Decrease	No Effect	
SEPS [71]	Increase		Decrease	
MSCM [48]	Decrease		Increase	
COCOMO II [17]	Increase		No Effect	
SPARS [47]	Both (usually increase)	Decrease		
Pfahl [106]	Both			
RDM [117]	Decrease	Decrease		
Rahmandad and Weiss [112]	Increase	Decrease		
Cao et al. [22]		Decrease		
Van Oorschot et al. [137]		Decrease	Decrease	

a negative relationship between knowledge transfer and schedule pressure, meaning those projects under study which reported knowledge transfer had less success staying in schedule and vice versa.

According to Staudenmayer et al. [128] temporal shifts can be used as coordination mechanisms enabling organizational change. Temporal shifts consist of variations in five dimensions of how people experience time: a sense of time pressure, sense of ability to allocate time to different activities, perceived tension among competing task demands, the time horizon considered, and sense of found time. An introduction of buffer time during a project allowed for better review and reassessment of the project and subsequently for more organizational change.

6.3.3. Effect on decision making

The 19-month ethnographic study conducted by Perlow et al. [104] showed that although fast decision making was initially beneficial for the company's growth, it later led to bad decisions. With a growing artificial sense of emergency, decisions had to be made faster, and the decision-making board ignored valid objections because making a fast decision had become more important than making the right decision.

The effect of time pressure on decision making in Agile software development was pondered at length by Riordan et al. [115]. In their conceptual framework, decision quality was explained with three temporal parameters: time pressure, polychronicity (i.e., unexpected or sporadic order of tasks), and iterative decision making.

Lohan et al. [72] investigated decision quality under time pressure. Better decision quality was achieved when time pressure was perceived to be stimulating, enjoyable, or satisfying. However, based on results when time pressure is perceived to be annoying, discouraging, and upsetting, there does not appear to be an effect on decision quality.

6.3.4. Effect in software process simulation models

During our literature search, we included papers examining process simulation models and their schedule compression effects. Table 8 presents the assumptions on the effects of time pressure underlying the models.

Two early studies [53,61] compared empirical data with established cost estimation models taking schedule compression or extension into account, namely COCOMO I [18] and Putnam's [111]. Jeffery [53] observed that, depending on the case, effort can either increase or decrease among 47 projects. Similarly, Kitchenham [61] noted two different schools of thought at the time. One where compressing or extending the schedule increases effort (COCOMO I), and one where compression increases effort and extension decreases it [111]. Because of varying results, Kitchenham reports that all underlying assumptions are likely invalid.

Smith et al. [124] added four task assignment factors to COCOMO I [18] and produced more accurate estimations on a single project. One of them, *intensity*, is defined as "the ratio between the number of active time units and the total number of time units during the development span". In the study, development effort has a negative relationship with intensity. This means that when development is focused on a single module, the overall effort on it decreases.

Later, the effects of different schedule compression levels on the estimates of COCOMO II [17] have been compared to real schedule compression ratios [51,144]. Yang et al. [144] also present an overview of other cost estimation models and their schedule compression approaches (PRICE-S PRICE Systems [109], SEER-SEM Fischman et al. [37] and SLIM Panlilio-Yap [96]). In COCOMO II compression increases the total effort needed for project completion. Based on newer empirical data, Hussain et al. [51] found an increased effort in highly compressed schedules.

Based on interviews, Rahmandad and Weiss [112] created a simulation model of the dynamics in concurrent software development. Developers under time pressure work harder, but also start to omit requirements, code reviews, unit testing, and documentation. This deteriorates quality and increases maintenance effort. In addition, organizational ability to produce software was reduced due to time pressure because not enough time was invested in improving the development tools or processes.

Lin et al. [71] introduce a simulation model based on literature reviews, interviews and expert reviews. If the initial schedule estimation is compressed, it results in an overall increased effort. Otherwise an extended schedule corresponds to slightly lessened effort. Hu et al. [48] produced a cost model based on a theory called Minimum Software Cost Model (MSCM), where the overall cost in man-months, gets higher with longer schedules, other things being equal.

Pfahl [106] created multiple models by conducting interviews and participating in review meetings. In one of them, increased schedule pressure leads to defect injection, and faster work progress. Overall, the effect of time pressure can result in earlier completion of the project, but also in delays when errors are introduced because of schedule pressure.

An Agile process simulation model was created based on interviews and an extensive literature review by Cao et al. [22]. Decreases in available time should result in adjustments in the scope or schedule, otherwise schedule pressure increases. Interviewees also stated that code refactoring was largely ignored and unit testing was reduced under schedule pressure, while pair-programming was said to perhaps alleviate the corner-cutting effects of pressure, as paired developers are usually more disciplined. No link between schedule pressure and development speed was suggested.

Another software process simulation model was developed by Abdel-Hamid and was discussed in multiple articles [1–3]. The model is based on interviews and software managers' review of the resulting model. The main model proposed that schedule pressure leads to process losses

and increases the error rate. However, the model does not have a link between schedule pressure and development rate, meaning it does not support the claim that schedule pressure improves development speed.

Houston et al. [47] simulated six risk factors of software development and produced a model called Software Project Actualized Risk Simulator (SPARS). The model assumed that the effects of excessive schedule pressure are fluctuating productivity, exhaustion, higher error creation, morale change, and weaker reviews. Based on previous models, Ruiz et al. [117] present a reduced dynamic model (RDM) where schedule pressure increases errors and productivity.

Van Oorschot et al. [137] validated a software process simulation model in a new product development (NPD) software project, according to which schedule pressure increases errors, overwork, task rejection, and delays. The authors simulate an actual project which suffered from schedule pressure, and show that with a later due date, the overall effort would have been decreased.

In **summary**, most simulation and cost models report an increase in effort when the project's schedule is compressed, see Table 8. While schedule compression can make developers work faster in the short term, it usually result in them omitting or avoiding reviewing, testing or documentating their code. This leads to an overall decrease in quality, more introduced errors and bugs, and eventually an increase in maintenance effort.

6.3.5. Effect on software process improvement

Several sources mentioned time pressure to be an essential obstacle in software process improvement (SPI) [8,9,99]. However, practitioner surveys differed on the importance of time pressure as a barrier, with the more recent paper ranking time pressure lower in importance [93,94].

Paulish and Carleton [99] recommend using SPI techniques to improve the process to meet future deadlines without emergencies. Similarly, Baddoo et al. [9] cite the lack of time as the biggest obstacle to software process improvement by participants from entry-level positions while strategic- and operational-level managers did not see the same importance for time pressure. Baddoo and Hall [8] report, based on a study of practitioner focus groups, that although both managers and developers reported the demotivating effect, the occurrence of time pressure as a demotivator was higher in focus groups composed of developers.

Niazi et al. [94] identify time pressure as a barrier to implementing software process improvement (only 17% of the time in interviews vs. 36% in scientific literature, rank 2 vs. rank 5). In a later article [93], time pressure as a barrier to SPI is mentioned again. Resources should be explicitly allocated to SPI efforts to ensure adequate time to complete tasks.

6.3.6. Effect on user involvement in the development process

Clegg et al. [27] present three case studies on software development in companies, one of which tried to include users in its software development process. An increase in time pressure decoupled the user and developer interaction, excluded user knowledge from the development process, and thus in the end, wasted resources and effort. However, afterwards the developers stated that involving users in development could be improved with more realistic deadlines and better management.

6.3.7. Effect on quality assurance

Baskerville et al. [12] reported that time pressure lowers quality of the code during the initial product development and leading to rework and redesign during later product development. Similarly, high time pressure caused by unrealistic deadlines leads to minimal quality assurance [139]. The quality assurance effort can be saved by using workarounds or compromises during implementation, by reducing the effort spent on documentation, by reallocating tasks to newly assigned developers, or by reducing the quality of the final product [14]. However, while the degree of time pressure seems to greatly reduce quality, [43] did not find an increased amount of defects on project with expedited schedules.

6.3.7.1. Effect on reviews.

In a broad survey [26] about software reviews with 226 respondents from companies of different sizes and from different countries, time pressure was cited the most often (75% of the time) as an obstacle to software review.

Staudenmayer et al. [128] studied change in the development cycle in a big software company. The company introduced *buffer times*. After several weeks of regular development, a buffer period of unallocated time is added. During the buffer time period, coding activity is suspended, but the tasks to be completed are not specified. Buffer time allows teams to reflect on the past and present, allowing the developers to switch from a development mode to a mode of reflection, awareness, and analysis. Developers can cope better with new or altered requirements caused by unexpected events, changes in customer needs, or other problems or ideas discovered during development. Teams in which buffer times were introduced kept to the schedule better and met release dates.

6.3.7.2. Effect on pair programming.

A study of pair programming of 31 developers from 4 companies [107] showed that time pressure (e.g., near the release date) leads developers to avoid pair programming and work individually to increase productivity.

6.3.7.3. Effect on testing.

In an experiment by Mäntylä et al. [81], a group under time pressure found fewer defects, but the difference was not statistically significant. Overall time pressure was associated with higher efficiency (more defects found per unit of time). However, Deak et al. [30] report a negative impact of time pressure on product quality, as well as its presence in Agile context when the project is behind schedule, as in the waterfall model.

6.4. Effect on outcome - time-cost-qualityscope

The so-called project management triangle dating back to the 1950s suggests that the outcome of any project can be explained by four constraints: quality, schedule, scope, and cost [6]. Time pressure in a project can be understood as a situation in which the project members realize that time, which can be either schedule (schedule pressure) or cost (time budget pressure), is running out. The project members try to avoid scheduling slippage or cost extension with various strategies, but typically by working faster. Studies have addressed quality, schedule, scope, and cost concerning time pressure. We divide the papers based on the data they use. We begin with studies with industrial project management data in Section 6.4.1, followed by experiments, surveys, and case studies in Sections 6.4.2 through 6.4.4. The results are summarized in Table 9.

For papers with quantitative data, we report whether time pressure has a positive or negative effect and statistical significance. We do not set an arbitrary threshold for significance reporting, often $p = 0.05$, as we consider this to be misleading. For example, if many sources all had $p = 0.06$, this would provide solid support for the impact of time pressure. Thus, omitting information based on an arbitrary threshold would not show this evidence.

6.4.1. Industrial project management data

Investigations of Capability Maturity Model (CMM) level 5 projects concerning effort, cycle time, and quality showed that schedule pressure decreases effort ($p = 0.065$) and cycle time ($p = 0.15$) [4]. In both cases, time pressure was the second most statistically significant predictor of the nine studies after project size. Schedule pressure was also linked to a decrease in quality but with a low p value ($p = 0.4$).

Hale et al. [43] studied among other things, whether software projects with expedited schedules had more defects than projects with non-expedited in a Capability Maturity Model Integration (CMMI) level

Table 9
Summary on the effects of time pressure on efficiency and quality. Sign (+, − or *U*) expresses the direction of relationship, i.e. whether time pressure increases (+) or decreases (−) efficiency improvement or quality reduction, or whether they have non-linear inverted *U*-shaped relationship (*U*).

Paper	N	Data	Pressure Condition	Efficiency Improvement	Quality Reduction
[4]	31	company projects	schedule	+, $p = 0.065$ (effort) +, $p = 0.15$ (cycle time)	+, $p = 0.4$ (defect count)
[43]	991	company projects	schedule	NA	−, $p = 0.4$ (increased amount of defects)
[88]	58	company projects	schedule	+, $p = 0.0001^{***}$ (effort)	NA
[66]	530	company projects	schedule	−, $p < 0.01^{**}$ (cost)	+, $p < 0.01^{**}$ (client satisfaction)
[91]	66	company projects	budget	*U*, $p = 0.05^{*}$(cycle time) *U*, $p = 0.01^{**}$ (effort)	−, $p = 0.00$ (defect / size) correlation only
[91]	66	company projects	schedule	+, $p = 0.01^{**}$ (cycle time) +, $p = 0.07$ (effort)	+, $p = 0.00$ (defects / size) correlation only
[90]	NA	company projects	schedule	*U*, $p < 0.01^{**}$ (cycle time) *U*, $p < 0.05$ (effort)	?, p = not significant
[23]	209	company projects	concurrent tasks	NA	+, $p < 0.01^{**}$ (defects)
[24]	1195	features in a project	time spent	NA	+, $p < 0.01^{**}$ (defects)
[81]	97	student experiment in test case development	reward	+, $p < 0.001^{***}$ (test case score / time)	−, $p = 0.922$ (test case score)
[81]	97	student experiment in requirement review	reward	+, $p = 0.002^{**}$ (defects found / time)	+, $p = 0.342$ (defects found)
[78]†	130	student experiment in manual testing	time-restriction	+, $p <$ 2.2e-16*** (defects / time)	+, $p =$ 2.96e-09*** defects found)
[35]	106	student experiment in software acquisition	time-restriction	+ p=0.020** (fixation duration)	+, $p = 0.013^{**}$ (number of fixations) +, $p = 0.344$ (number values) −, $p = 0.957$ (number of labels)
[113]	100	experiment software maintenance task	reward	+, $p = 0.0399$ (time used, 2x2 Anova)	NA
[133]††	60	experiment db query	time restriction	−, $p = 0.5138$ (correct per minute, simple) −, $p = 0.8243$ (correct per minute, complex)	+ p=0.0001*** (correct, simple task) + $p = 0.1281$ (correct, complex task)

$^{*}p$-values $<$ 0.05 highlighted with *, $<$ 0.01 with ** and $<$ 0.001 with *** † *p*-values missing from original paper computed from raw for this paper †† we performed t-test from reported group means and standard deviations between low and high time pressure groups

3 company. However this hypothesis concerning schedule pressure was not supported ($p = 0.400$). Mukhopadhyay and Kekre [88] investigated 58 software projects in the process control manufacturing domain and found that schedule pressure decreased software project effort with high statistical significance ($p = 0.0001$), while other statistically significant predictors were project size and programmer speed.

Langer et al. [66] investigated the practical intelligence of project managers. The evidence showed that difficult projects achieved better client satisfaction and cost performance than standard projects because of the practical intelligence of the project manager. Concerning direct effects, the study reported that schedule pressure, surprisingly, increases cost ($p < 0.01$) and reduces client satisfaction ($p < 0.01$) even when they are controlled for project size.

Nan and Harter [91] investigated the inverted *U*-shaped relationship (the initial pressure improves, but after a certain point, the pressure decreases performance) of budget and schedule pressure of 66 projects. They used regression models to predict the cycle time and development effort using budget and schedule pressure while controlling for the process maturity, size, complexity, and quality of the projects. For budget pressure, they found a *U*-shaped relationship. Furthermore, the linear terms of schedule pressure were negative, meaning an increase in schedule pressure reduced the cycle time and development effort. From the paper appendix, we found that quality had a negative correlation with budget pressure (−0.51), but a positive correlation with schedule pressure (0.51).

In earlier work, Nan et al. [90] conducted a similar investigation in a large company ($1 billion/year). They found that time pressure (schedule) had a *U*-shaped relation with cycle time or effort. They also found that pressure had a non-statistically significant relationship with quality. However, this earlier work omitted many details, such as sample size and did not show the statistical values, making the results less trustworthy.

Cataldo [23] reports that time pressure measured as concurrent execution of tasks was the most important source of errors ($p < 0.01$) in distributed software development projects. In the regression model, the expected number of defects increased by 47.1% when the value measuring time pressure changed from the minimum to maximum value. Similarly, Cataldo and Herbsleb [24] report that in global feature-oriented software development, time was the most significant factor when feature integration failed ($p < 0.01$), as well as associated with a lower likelihood of failures in that feature. In the regression model based on a project with 1.5 million lines of code and 1195 features, an additional week corresponds to a lower likelihood of 0.8% of integration failure.

6.4.2. Experiments

A controlled experiment in requirements review and test case development Mäntylä et al. [81] showed that time pressure reduced effectiveness in defect detection ($p = 0.342$) and had no impact on test case quality ($p = 0.922$). Efficiency, effectiveness divided by time, improved defect detection ($p = 0.002$) and test case quality ($p < 0.001$). This result supports the idea that effectiveness (or quality of the work) will decrease, but efficiency will increase due to less time being used.

An experiment on time pressure in manual testing by Mäntylä and Itkonen [78] also showed lower effectiveness ($p < 0.001$) and higher efficiency ($p < 0.001$) under time pressure. The paper reports no t-test so we computed the *p*-values from the raw data. The researchers also concluded that combining several time-pressured testers would have been beneficial because "we can either find roughly the same amount of defects with 59% less effort, or we can use the same effort to find 71% more defects." The drawback of using several independent time-pressured testers would have been the extra work in duplicate defect filtering.

In another experiment, Fehrenbacher and Smith [35] performed an experiment about time pressure in software acquisition. The study shows

that under time pressure individuals worked faster but felt less confident in their decisions and were keener to postpone it. Gaze duration was reduced under time pressure as the individuals try to work faster ($p = 0.02$). Under time pressure, the work strategy is focused on higher-level topics, while on the other hand, without time pressure, more effort is spent looking at the details ($p = 0.013$). The best search of information for the software acquisition task occurred under time pressure and with requirements for explicitly written reasoning about the acquisition choice. Hence according to results, to achieve the best performance regarding effectiveness and time spent, time pressure and a quality control requirement should be used.

Lee et al. [68] showed in an experiment that project escalation, that is, willingness to continue a troubled software project, is less likely to happen if there is high time pressure in the project. This willingness to stop a problematic project was generally seen as positive by the authors as project escalation can waste valuable resources in "a failing course of action".

An experiment by Ramanujan et al. [113] investigated the time used in short software maintenance tasks. For documented programs time pressure had no effect, but for programs with no documentation time pressure reduced maintenance task time by 20%. The authors also found that the performance of participants with low and high knowledge increased ($p = 0.0399$) under time pressure. However, the impact was larger for low-knowledge participants (16%) than for high-knowledge participants (6%). The results offer further support that time pressure reduces effort. However, the finding that low-knowledge participants are affected more by time pressure is contrary to previous research.

Another controlled experiment by Topi et al. [133] of database query development tasks showed unexpectedly that time pressure didn't make the subjects work faster and reduced their efficiency. However, the p-values were high ($p = 0.51$ simple task, $p = 0.82$ complex task). Effectiveness was reduced under time pressure as well ($p < 0.001$ simple task, $p = 0.1281$ complex task). The authors provide the following explanation: "The subjects did not have good mechanisms for accelerating their work. Thus, this seems to indicate that with this task type, just reducing the available time does not improve productivity." On the other hand, time pressure reduced the number of total correct database queries.

6.4.3. Surveys

Investigation of software project teams reports a statistically significant inverted *U*-shaped relationship between team process and time pressure as the quadratic term of time pressure Maruping et al. [83]. The authors did not provide separate measures for product quality, or the effort used, but combined them in a team performance metric. However, temporal leadership statistically significantly removed the *U*-shaped relationship so that with strong temporal leadership, only positive effects of time pressure on team process exist. In the paper, temporal leadership was defined as "the structuring, coordination, and management of task pacing in teamwork."

Lohan et al. [72] investigated the effect of group cohesion, perceived challenge, and hindrance time pressure on decision-making quality of information system development. Challenge time pressure perceived as stimulating, enjoyable, and satisfying was thought to have a positive effect on quality. However, perceived hindrance time pressure did not have a negative effect.

In a survey, Nugroho and Chaudron [95] found that meeting a deadline was considered by the respondents to be the smallest factor driving deviations between the design and code with 27% of responses stating that the deadline never caused deviations. Of the chosen factors, meeting a deadline was the only factor that did not directly mention design quality (other factors being impractical design, incomplete design, and design not satisfying the requirements).

A survey conducted by Verner et al. [139] on software project failures found that too aggressive delivery dates caused time pressure, which then caused the omission of QA practices and led to project failures in six out of the eight projects studied. The authors proposed that projects should be kept short and manageable to prevent failures from too aggressive delivery dates. This practice sounds like Agile with small iterations. Another survey by Ferreira et al. [36] showed that requirements volatility causes time pressure which increases errors in generating requirements.

6.4.4. Case studies

A qualitative case study by Lavallée and Robillard [67] in a telecom company provided a concrete example of how taking shortcuts reduces overall quality. The researchers found that budget pressure prevented the implementation of a proper company-wide solution to a technical problem and resulted in a cheap patch solution that was repeated by at least 12 development teams. Each team was protecting their budget and decided to take a shortcut solution. A case study in another telecom company showed that time pressure was selected as the root cause for 40% of the defects [70]. A more detailed investigation showed that for algorithmic defects, the share of time pressure was as high as 70%, while for functionality defect type it was only 17%. The authors elaborated that "functionality defect refers to missing or wrong functionality (w.r.t. requirements) and algorithm defect refers to an inadequate (efficiency) or wrong (correctness) algorithmic realization."

6.4.5. Summary

We summarize quantitative empirical evidence from analysing software companies and software engineering experiments that performed statistical tests regarding whether time pressure improves development efficiency and whether time pressure reduces development quality. For this, we consider only papers that measured actual outcomes. Thus, we omit the results of questionnaire surveys. Development efficiency means that software development is faster in terms of the cycle time, or effort, or both. In experiments, development efficiency means that effectiveness per time unit is faster, *e.g.*, more defects per hour are found. Quality reduction in company cases was often measured by the number of defects or customer satisfaction. In experiments, quality reduction, becomes effectiveness, *e.g.*, fewer defects are found in reviews, or less correct database queries are made.

Table 9 shows the results. The + sign means that the paper found an impact in the direction predicted in the column headings, that is, improved efficiency or reduced quality. The - sign means the opposite, and *U* sign means that some pressure results in the predicted impact, but too much pressure causes the opposite effect. This inverted *U*-shape effect comes from the Yerkes-Dodson law which states that initial pressure improves performance while pressure increasing above a certain point reduces the performance. After the sign, we report the statistical significance from the paper. The statistical significance can originate from various statistical tests, such as regression models, correlations, or t-tests.

Seven papers support improvement in development efficiency due to time pressure, two papers report inverted *U*-shaped results, and two papers report a decreased efficiency. One of the two papers offering the counter-evidence had a strong statistical significance [66] but offered no explanation. Therefore we contacted the authors for further details but received no response. In the other paper, the statistical significance was much lower: $p = 0.5$ and 0.8. The paper also offered an explanation: in the case of database development tasks, the subjects were unable to go faster. Overall, we conclude from quantitative analysis of company data and experiments, that small to medium size time pressure is beneficial for development efficiency (9 paper for and 2 papers against). This is partly in contradiction to Section 6.3.4, where majority of models assume the total amount of effort to increase with schedule compression.

Nine papers support a reduction in quality due to time pressure. Four papers report the opposite, that time pressure increased quality, and two papers provide no data on quality reduction. If we examine the

three papers offering the counter-evidence, the statistical significance was very low on three papers $p = 0.922$ [81], p = 0.4 [43] and $p = 0.957$ [35], while in the remaining paper [91] with high statistical significance ($p = 0.00$) quality improvement computation was shown with a correlation only with industrial data. A regression model or partial correlation controlling for confounding factors would be a more robust alternative. We conclude from qualitative analysis of company data and experiments, that time pressure reduces quality in software engineering, but we suspect this is because less time is available or used. Reduced quality due to time pressure is further supported unanimously by the cost and simulation models in Section 6.3.4.

7. Threats to validity

The first threat to the validity of the findings is the search strings we used to query search engines for the literature. Before starting the literature review, we familiarized ourselves with the topic and iteratively improved the search strings. In total, we used synonyms and different ways of spelling, but it is possible some sources could have been missed with the otherwise inconsistent terminology used in the literature. However, as we did not run into other terms in the snowballing and analysis phase, we believe we have covered at least the terms most frequently used in the literature.

Several databases and search engines can be used to search scientific literature. Due to the limited amount of resources, we decided to use those that had the most extensive coverage, namely Scopus and Google Scholar. From these, partial automatic data retrieval is possible. However, it is possible that more sources of information could have been found by searching other academic databases, as data in a single one is incomplete and can contain errors such as missing abstracts. Indeed, we are aware of missing abstracts on some conference proceedings related to information systems. Another limiting factor in our study was using only the first 100 search results for Google Scholar searches, which were meant to complement the primary searches made with Scopus. Indeed, more sources could possibly be found by increasing the number of search results examined for Google Scholar. As our resources are limited, and the application of the selection criteria more laborious than usual with reading at least the abstract, the boundary has to be set somewhere. We have tried to combat these issues with conducting backwards and forwards snowballing.

Although we applied the selection criteria specified in Section 3.1 when we identified the relevant literature, we may have missed relevant papers. This is more the case in studies where time pressure was not the main topic investigated but constituted some of the empirical evidence. There is an inherent trade-off between the effort spent and the number of details that can be examined in the papers while applying the selection criterion.

The inexperience of the first author on performing systematic reviews can be mentioned as a threat to validity. However, other authors had previous experience on conducting systematic literature reviews and they provided guidance. First author had gained experience on conducting the previous work [63]. Similarly, reviewer bias in study selection could be an issue. We tried to solve this as best as we could by marking even remotely borderline cases up for discussion as explained in Section 3.

We could could have missed some details in the qualitative coding phase with NVivo due to errors in this stage. We are also aware of some more recent papers on the topic that were published during the analysis of the collected literature [64,118]. However, adding the most recent papers to the analysis would be a never-ending circle. Last, we want to mention publication bias as a threat to these findings [129]. It can be formulated that results, where no links between time pressure and investigated processes or approaches are found, might have a smaller chance of being published.

8. Conclusions

We conclude the paper by first highlighting the contributions of this work. Next, we provide practical takeaways, and we outline directions for future work.

8.1. Contributions

In this article, we perform the largest literature review related to time pressure in software engineering. Our main contributions are as follows.

- In Section 4.1 we provide the definitions of time pressure as used in software engineering literature. They can be roughly be divided into two categories. The first one is based on the Yerkes-Dodson law, which states that the amount of time pressure affects performance in an inverted *U*-shaped form. Initial increases in pressure improve performance, although only up to a certain point, after which further increases in time pressure decrease performance. The second category of definitions is based on the challenge-hindrance framework, which states that positive (challenge) time pressure improves performance, while negative (hindrance) decreases performance.
- In Section 4.2, we provide a list of papers containing metrics and operationalizations, used in previous literature to measure and operationalize time pressure in software engineering.
- In Section 5.2, we map the selected papers to different stages of the software development process and approaches (see Table 7). The main topics of papers related to time pressure were found to be either cost estimation or quality assurance.
- In Section 6.1, we summarize the reported causes of time pressure: problems in effort estimation, scheduling, commercial pressures, management styles, and social settings.
- We review the effects of time pressure on individuals and software process outcomes. We summarize the corresponding quantitative results in Table 9. The predominant effect of time pressure on outcome is that it reduces quality while increasing efficiency. However, many of the cost and process simulation models in Table 8 support the increase in overall effort for a software project with compressed schedule.

8.2. Practical takeaways

We have performed many primary studies ourselves on performance under time pressure and arousal detection (e.g., [64,78]). After spending considerable effort on systematically familiarizing ourselves with related literature, we want to conclude this paper by providing a practitioner-oriented summary with key takeaways.

Time pressure is common in the software industry, and it can be caused by commercial pressure, company culture, or errors in effort estimation (see Section 6.1). Time pressure can have both positive and negative outcomes. On the positive side, it increases efficiency in the short term: the sense of urgency that time pressure creates provides focus on the basic product requirements. On the negative side, the lack of time reduces the quality of the outcome, leads to tunnel vision, and limits opportunities for improving the software product and process. See Section 6.4 and Table 9 for details.

The question thus becomes: “can a software project achieve the best of both worlds: increased efficiency and urgency while avoiding reduced quality?” The answer is yes and no. No, in the sense that the best of both worlds cannot be achieved simultaneously. With heavy time pressure, it is difficult to find the time to make important improvements to the product during the software development process. However, it is possible for a project to have periods of time pressure and periods of buffer and reflection time, in which the former provides efficiency while the latter ensures a high quality of the product and process. This is a balance that skillful software engineers and managers should aim to achieve.

The amount and type of time pressure also play a role (see Section 4.1). Small to moderate time pressure brings out positive effects, while very high time pressure provides no additional benefits. If the pressure is experienced as positive, it leads to a more positive outcome than if the time pressure is experienced as negative. Positive time pressure can be achieved when a team feels that timely delivery is essential. Conversely, multiple conflicting goals, such as having to deliver a high-reliability product with minimal effort, can generate a negative time pressure. Typically, negative time pressure is more intense and has the trait of impossibility attached to it. Software engineers and managers should be mindful how the software development team feels about the time pressure they are dealing with.

Saying that time pressure improves efficiency and reduces quality in software engineering is a generalization. Important task dependent variations are likely to occur. Although empirical evidence of task-dependent effects of time pressure in software engineering is limited, we found partial evidence for two variations. First, it appears that time pressure most often occurs during software quality assurance and testing (see Section 5.2). This happens because testing, in particular, is the last phase that precedes software release: therefore, all the schedule slips of earlier phases are felt during software testing. Another possible cause is having a lower quality product because of time pressure, which ends up needing more testing. Second, the effects of time pressure vary according to the type of tasks. We found evidence that tasks with a high algorithmic nature have fewer efficiency improvements and suffer more from reduced quality than other types of tasks under time pressure [70,133]. Software engineers and managers should be particularly mindful in ensuring that software testing is not under too much negative time pressure and that tasks that are highly algorithmic in nature are under minimal pressure.

When it comes to different software process models, there is some empirical evidence and theoretical reasoning why Agile software development and time pressure are a good fit. We believe there are three reasons for this. First, in Agile software development, iterations are small, meaning that there is a constant low time pressure to meet the next deadline, although there is no massive final deadline [44,134]. The sustainable pace of Agile is also good at cutting down extreme time pressure. Psychological experiments have shown that aggressive intermediate deadlines can improve outcome quality and individuals' time management [5]. We assume the same is true for software development teams. Second, in Agile software development, quality assurance and testing go hand-in-hand with development and thus, avoid the intense time pressure that haunts software testing in traditional phases. However, Agile is not a silver bullet, as lack of testing and re-factoring have been reported in Agile projects as well [22,30]. Third, team empowerment, which is higher in Agile than in traditional projects, can block the negative stress caused by time pressure [65]. This effect can be linked to the well-established occupational theory based on the job demands-control model [10,57]. This model proposes that adverse effects of time pressure (and other stressors) can be reduced when an employee has high independence and decision latitude in the job. This is precisely the case for Agile teams with high empowerment.

In more traditional development, processes can suffer from the effects of one final deadline, especially if there are no intermediate deadlines. Focusing on one final deadline can make the workload uneven, which has been linked to integration failures [24]. Similarly, time budget pressure, combined with each team optimizing their own project, has been reported to lead to multiple teams developing a cheap patch instead of a company-wide solution [67].

8.3. Takeaways for researchers

There seems to be a contrast in results on productivity under time pressure between empirical studies on project performance and studies creating cost and process models, as it can be seen in Tables 8 and 9. The majority of cost and process models assume that overall effort increases with compressed schedules, whereas most empirical studies report improved efficiency under time pressure. There are multiple possible explanations for this. For example, in some of the models, the increased effort in compressed schedules is the result of an increase in error rate; thus, increased maintenance is needed to complete the project. Moreover, the time scale is also important: improved efficiency is more likely to occur in the short term, whereas negative effects become pronounced in the long term. Future studies could help establish guidelines for these trade-offs.

Because of the *U*-shaped relationship between performance and arousal based on the Yerkes-Dodson law, quantifying the amount of time pressure on an individual would help investigate its effects at the project level data. The multiple assumptions and effects of schedule compression, as noted in Table 8, are can be partially explained by the difference in contexts in which software projects have been developed. However, the effects of time pressure on individuals might also play a role. Indeed, as it can be seen in Table 9, quantitative studies focused on experiments either on individuals, or on groups at the project level, but not on both at the same time. From the developers' point of view, it should matter if the system is developed with less re-usable code or with more overtime hours. This is further indicated by recent studies using the challenge-hindrance framework [69]. Taking into account these effects in future quantitative studies would provide better context and reasoning for conflicting results.

As noted in Section 6.1, prior literature has identified company culture as a cause of time pressure. In these types of situations, hurry and time pressure are either prolonged or constant [54,102]. The negative effects of time pressure, such as increased stress, burnout, and depression, are reported to be products of exhaustion and job strain. Hence, the negative effects of time pressure might not become apparent in a single software project and especially in a lab experiment, if time pressure is otherwise at manageable levels or even rare in the company. We believe that company culture, together with varying time scales, might explain the discrepant assumptions and effects of schedule pressure and compression, noted in Tables 8 and 9.

Many recent studies have tried to detect time pressure with various techniques, including sentiment analysis [80], repository mining [64], and physiological measurements [62,134]. However, while promising results have been acquired, none of them have been able to reliably detect time pressure within a single project. Hence, if possible, future work should aim to accomplish this through combining different data sources. It could eventually provide project managers with up-to-date information on the state of a project, as felt by the individual developers.

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.

Acknowledgements

This work has been supported by Academy of Finland grant 298020. The first author has been supported by Kaute-foundation.

References

[1] T.K. Abdel-Hamid, Investigating the cost/schedule trade-off in software development, IEEE Softw. 7 (1) (1990) 97–105.
[2] T.K. Abdel-Hamid, S.E. Madnick, Lessons learned from modeling the dynamics of software development, Commun. ACM 32 (12) (1989) 1426–1438.
[3] T.K. Abdel-Hamid, S.E. Madnick, The elusive silver lining: how we fail to learn from failure in software development, Sloan Manage. Rev. (1990).
[4] M. Agrawal, K. Chari, Software effort, quality, and cycle time: a study of CMM level 5 projects, IEEE Trans. Softw. Eng. 33 (3) (2007).
[5] D. Ariely, K. Wertenbroch, Procrastination, deadlines, and performance: Self-control by precommitment, Psychol. Med. 13 (3) (2002) 219–224.
[6] R. Atkinson, Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria, Int. J. Project Manage. 17 (6) (1999) 337–342.

[7] R.D. Austin, The effects of time pressure on quality in software development: an agency model, Inf. Syst. Res. 12 (2) (2001) 195–207.
[8] N. Baddoo, T. Hall, De-motivators for software process improvement: an analysis of practitioners' views, J. Syst. Softw. 66 (1) (2003) 23–33.
[9] N. Baddoo, T. Hall, D. Wilson, Implementing a people focused SPI programme, in: Proceedings of 11th European Software Control and Metrics Conference Munich, 2000.
[10] A.B. Bakker, E. Demerouti, The job demands-resources model: state of the art, J. Managerial Psychol. 22 (3) (2007) 309–328.
[11] R.D. Banker, C.F. Kemerer, Factors affecting software maintenance productivity: an exploratory study, in: Proceedings of the 8th International Conference on Information Systems, Pittsburgh, PA, USA, AISEL, 1987, p. 27.
[12] R. Baskerville, L. Levine, J. Pries-Heje, S. Slaughter, How internet software companies negotiate quality, Computer 34 (5) (2001) 51–57.
[13] D. Basten, The role of time pressure in software projects: a literature review and research agenda, in: eProceedings of the 12th International Research Workshop on Information Technology Project Management (IRWITPM), 2017, pp. 1–15.
[14] D. Basten, W. Mellis, A current assessment of software development effort estimation, in: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, IEEE, 2011, pp. 235–244.
[15] E. Bjarnason, K. Wnuk, B. Regnell, Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inf. Softw. Technol. 54 (10) (2012) 1107–1124.
[16] J.D. Blackburn, G.D. Scudder, Time-based software development, Integr. Manuf. Syst. 7 (2) (1996) 60–66.
[17] B. Boehm, C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D.J. Reifer, B. Steece, Cost Estimation with COCOMO II, ed: Upper Saddle River, NJ: Prentice-Hall (2000).
[18] B.W. Boehm, Software Engineering Economics, vol. 197, Prentice-Hall Englewood Cliffs (NJ), 1981.
[19] A. Borg, A CIAR Study in a Male Dominated ICT Company in Malta Which Looks at Work-Life Issues Through the Masculine Lens: A Case of: If it ain't Broke, don't fix it? Ph.D. thesis, Middlesex University, 2014.
[20] F.P. Brooks Jr., The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, 2nd ed., Pearson Education India, 1995.
[21] M.A. Campion, R.G. Lord, A control systems conceptualization of the goal-setting and changing process, Organ. Behav. Hum. Perform. 30 (2) (1982) 265–287.
[22] L. Cao, B. Ramesh, T. Abdel-Hamid, Modeling dynamics in agile software development, ACM Trans. Manage. Inf. Syst. (TMIS) 1 (1) (2010) 5.
[23] M. Cataldo, Sources of errors in distributed development projects: implications for collaborative tools, in: Proceedings of the 2010 ACM Conference on Computer-Supported Cooperative Work, ACM, 2010, pp. 281–290.
[24] M. Cataldo, J.D. Herbsleb, Factors leading to integration failures in global feature-oriented development: an empirical analysis, in: Software Engineering (ICSE), 2011 33rd International Conference on, IEEE, 2011, pp. 161–170.
[25] D.S. Chong, W. Van Eerde, K.H. Chai, C.G. Rutte, A double-edged sword: the effects of challenge and hindrance time pressure on new product development teams, IEEE Trans. Eng. Manage. 58 (1) (2011) 71–86.
[26] M. Ciolkowski, O. Laitenberger, S. Biffl, Software reviews, the state of the practice, IEEE Softw. 20 (6) (2003) 46–51.
[27] C.W. Clegg, P.E. Waterson, C.M. Axtell, Software development: knowledge-intensive work organizations, Behav. Inf. Technol. 15 (4) (1996) 237–249.
[28] C.L. Cooper, P.J. Dewe, M.P. O'Driscoll, Organizational Stress: A Review and Critique of Theory, Research, and Applications, Sage, 2001.
[29] S.H. Costello, Software engineering under deadline pressure, ACM SIGSOFT Softw. Eng. Notes 9 (5) (1984) 15–19.
[30] A. Deak, T. Stålhane, G. Sindre, Challenges and strategies for motivating software testing personnel, Inf. Softw. Technol. 73 (2016) 1–15.
[31] E. Demerouti, A.B. Bakker, F. Nachreiner, W.B. Schaufeli, The job demands-resources model of burnout., J. Appl. Psychol. 86 (3) (2001) 499.
[32] C.C. Durham, E.A. Locke, J.M. Poon, P.L. McLeod, Effects of group goals and time pressure on group efficacy, information-seeking strategy, and performance, Hum. Perform. 13 (2) (2000) 115–138.
[33] S. Easterbrook, J. Singer, M.-A. Storey, D. Damian, Selecting empirical methods for software engineering research, in: Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 285–311.
[34] C. Ebert, C. Jones, Embedded software: facts, figures, and future, Computer 42 (4) (2009).
[35] D. Fehrenbacher, S. Smith, Behavioural affect and cognitive effects of time-pressure and justification requirement in software acquisition: evidence from an eye-tracking experiment, in: 20th Americas Conference on Information Systems (AMCIS), 2014.
[36] S. Ferreira, J. Collofello, D. Shunk, G. Mackulak, Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation, J. Syst. Softw. 82 (10) (2009) 1568–1577.
[37] L. Fischman, K. McRitchie, D. Galorath, Inside seer-sem, in: CrossTalk: The, 2005, p. 146.
[38] Y. Fujigaki, Time series investigation of job-events and depression in computer software engineers, Ind. Health 34 (2) (1996) 71–79.
[39] Y. Fujigaki, K. Mori, Longitudinal study of work stress among information system professionals, Int. J. Hum. -Comput. Interact. 9 (4) (1997) 369–381.
[40] S. Gaudin, Silicon valley's 'pressure cooker:' thrive or get out, 2015, [Online; posted 18-August-2015] https://www.computerworld.com/article/2972723/it-careers/silicon-valleys-pressure-cooker-thrive-or-get-out.html.
[41] T. Gilb, S. Finzi, Principles of Software Engineering Management, vol. 11, Addison-wesley Reading, MA, 1988.
[42] D. Graziotin, F. Fagerholm, X. Wang, P. Abrahamsson, On the unhappiness of software developers, in: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, ACM, 2017, pp. 324–333.
[43] D.P. Hale, J.E. Hale, R.K. Smith, Evaluation of work product defects during corrective & enhancive software evolution: a field study comparison, ACM SIGMIS Database 42 (1) (2011) 59–73.
[44] M.L. Harris, R.W. Collins, A.R. Hevner, Agile methods: fast-paced, but how fast? in: AMCIS 2009 Proceedings, 2009, p. 149.
[45] S.G. Hart, Nasa-task load index (nasa-tlx); 20 years later, in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 50, Sage CA: Los Angeles, CA, 2006, pp. 904–908.
[46] R.P. Heitz, The speed-accuracy tradeoff: history, physiology, methodology, and behavior, Front. Neurosci. 8 (2014) 150.
[47] D.X. Houston, G.T. Mackulak, J.S. Collofello, Stochastic simulation of risk factor potential effects for software development risk management, J. Syst. Softw. 59 (3) (2001) 247–257.
[48] Q. Hu, R.T. Plant, D.B. Hertz, Software cost estimation using economic production models, J. Manage. Inf. Syst. 15 (1) (1998) 143–163.
[49] L. Huang, M. Endrawes, A. Hellman, An experimental examination of the effect of client size and auditors' industry specialization on time pressure in australia, Corp. Ownersh. Control 12 (4) (2015) 398–408.
[50] R.T. Hughes, Expert judgement as an estimating method, Inf. Softw. Technol. 38 (2) (1996) 67–75.
[51] S. Hussain, S.A. Khoja, N. Hassan, P. Lohana, Effect of schedule compression on project effort in COCOMO II model for highly compressed schedule ratings, in: International Multi Topic Conference, Springer, 2008, pp. 202–214.
[52] M.I. Hwang, Decision making under time pressure: a model for information systems research, Inf. Manage. 27 (4) (1994) 197–203.
[53] D.R. Jeffery, Time-sensitive cost models in the commercial MIS environment, IEEE Trans. Softw. Eng. (7) (1987) 852–859.
[54] D. Jemielniak, Time as symbolic currency in knowledge work, Inf. Organ. 19 (4) (2009) 277–293.
[55] C. Jones, Social and technical reasons for software project failures, CrossTalk 19 (6) (2006) 4–9.
[56] N. Juristo, S. Vegas, The role of non-exact replications in software engineering experiments, Empir. Softw. Eng. 16 (3) (2011) 295–324.
[57] R.A. Karasek Jr, Job demands, job decision latitude, and mental strain: implications for job redesign, Adm. Sci. Q. (1979) 285–308.
[58] K. Karhu, O. Taipale, K. Smolander, Investigating the relationship between schedules and knowledge transfer in software testing, Inf. Softw. Technol. 51 (3) (2009) 663–677.
[59] J.R. Kelly, J.E. McGrath, Effects of time limits and task types on task performance and interaction of four-person groups, J. Pers. Soc. Psychol. 49 (2) (1985) 395.
[60] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE-2007-01, 2007.
[61] B.A. Kitchenham, Empirical studies of assumptions that underlie software cost-estimation models, Inf. Softw. Technol. 34 (4) (1992) 211–218.
[62] A. Kołakowska, A. Landowska, M. Szwoch, W. Szwoch, M.R. Wróbel, Emotion recognition and its application in software engineering, in: Human System Interaction (HSI), 2013 The 6th International Conference on, IEEE, 2013, pp. 532–539.
[63] M. Kuutila, M.V. Mäntylä, M. Claes, M. Elovainio, Reviewing literature on time pressure in software engineering and related professions: computer assisted interdisciplinary literature review, in: Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering, IEEE Press, 2017, pp. 54–59.
[64] M. Kuutila, M.V. Mäntylä, M. Claes, M. Elovainio, B. Adams, Using experience sampling to link software repositories with emotions and work well-being, in: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2018, p. 29.
[65] M. Laanti, Agile and wellbeing–stress, empowerment, and performance in Scrum and Kanban teams, in: System Sciences (HICSS), 2013 46th Hawaii International Conference on, IEEE, 2013, pp. 4761–4770.
[66] N. Langer, S.A. Slaughter, T. Mukhopadhyay, Project managers' practical intelligence and project performance in software offshore outsourcing: a field study, Inf. Syst. Res. 25 (2) (2014) 364–384.
[67] M. Lavallée, P.N. Robillard, Why good developers write bad code: an observational case study of the impacts of organizational factors on software quality, in: Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on, vol. 1, IEEE, 2015, pp. 677–687.
[68] J.S. Lee, M. Keil, V. Kasi, The effect of an initial budget and schedule goal on software project escalation, J. Manage. Inf. Syst. 29 (1) (2012) 53–78.
[69] J.A. LePine, N.P. Podsakoff, M.A. LePine, A meta-analytic test of the challenge stressor–hindrance stressor framework: an explanation for inconsistent relationships among stressors and performance, Acad. Manage. J. 48 (5) (2005) 764–775.
[70] M. Leszak, D.E. Perry, D. Stoll, A case study in root cause defect analysis, in: Proceedings of the 22nd International Conference on Software Engineering, ACM, 2000, pp. 428–437.
[71] C.Y. Lin, T. Abdel-Hamid, J.S. Sherif, Software-engineering process simulation model (SEPS), J. Syst. Softw. 38 (3) (1997) 263–277.
[72] G. Lohan, T. Acton, K. Conboy, An investigation into time pressure, group cohesion and decision making in software development groups, in: Proceedings of the 25th Australasian Conference on Information Systems, Auckland, New Zealand, ACIS, 2014.

[73] I.S. MacKenzie, P. Isokoski, Fitts' throughput and the speed-accuracy tradeoff, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2008, pp. 1633–1636.
[74] C.R. Madan, M.L. Spetch, E.A. Ludvig, Rapid makes risky: time pressure increases risk seeking in decisions from experience, J. Cognit. Psychol. 27 (8) (2015) 921–928.
[75] O. Malgonde, R. Collins, A. Hevner, Applying emergent outcome controls to mitigate time pressure in agile software development, in: Proceedings of the Americas Conference on Information Systems, 2014, pp. 1–7.
[76] C. Mann, F. Maurer, A case study on the impact of scrum on overtime and customer satisfaction, in: Agile Conference, 2005. Proceedings, IEEE, 2005, pp. 70–79.
[77] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, M. Ortu, Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity? in: Proceedings of the 13th International Conference on Mining Software Repositories, ACM, 2016, pp. 247–258.
[78] M.V. Mäntylä, J. Itkonen, More testers–the effect of crowd size and time restriction in software testing, Inf. Softw. Technol. 55 (6) (2013) 986–1003.
[79] M.V. Mäntylä, F. Khomh, B. Adams, E. Engstrom, K. Petersen, On rapid releases and software testing, in: Software Maintenance (ICSM), 2013 29th IEEE International Conference on, IEEE, 2013, pp. 20–29.
[80] M.V. Mäntylä, N. Novielli, F. Lanubile, M. Claes, M. Kuutila, Bootstrapping a lexicon for emotional arousal in software engineering, in: Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on, IEEE, 2017, pp. 198–202.
[81] M.V. Mäntylä, K. Petersen, T.O. Lehtinen, C. Lassenius, Time pressure: a controlled experiment of test case development and requirements review, in: Proceedings of the 36th International Conference on Software Engineering, ACM, 2014, pp. 83–94.
[82] M. Marques, S.F. Ochoa, A. Quispe, L. Silvestre, A. Villena, Coordination and pressing: a formula for teamwork - a case study, in: Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on, IEEE, 2010, pp. 83–88.
[83] L.M. Maruping, V. Venkatesh, S.M. Thatcher, P.C. Patel, Folding under pressure or rising to the occasion? Perceived time pressure and the moderating role of team temporal leadership, Acad. Manage. J. 58 (5) (2015) 1313–1333.
[84] A.J. Maule, G.R.J. Hockey, L. Bdzola, Effects of time-pressure on decision-making under uncertainty: changes in affective state and information processing strategy, Acta Psychol. 104 (3) (2000) 283–301.
[85] S. McConnell, Rapid Development: Taming Wild Software Schedules, Pearson Education, 1996.
[86] E. Miranda, A. Abran, Protecting software development projects against underestimation, Project Manage. J. 39 (3) (2008) 75–85.
[87] K. Molokken, M. Jorgensen, A review of software surveys on software effort estimation, in: Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on, IEEE, 2003, pp. 223–230.
[88] T. Mukhopadhyay, S. Kekre, Software effort models for early estimation of process control applications, IEEE Trans. Softw. Eng. 18 (10) (1992) 915–924.
[89] S. Mullainathan, E. Shafir, Scarcity: Why Having Too Little Means So Much, Macmillan, 2013.
[90] N. Nan, D. Harter, T. Thomas, The impact of schedule pressure on software development: a behavioral perspective, in: ICIS 2003 Proceedings, 2003, p. 71.
[91] N. Nan, D.E. Harter, Impact of budget and schedule pressure on software development cycle time and effort, IEEE Trans. Softw. Eng. 35 (5) (2009) 624–637.
[92] M.H.N. Nasir, S. Sahibuddin, Critical success factors for software projects: a comparative study, Sci. Res. Essays 6 (10) (2011) 2174–2186.
[93] M. Niazi, Software process improvement implementation: avoiding critical barriers, CROSSTALK 22 (1) (2009) 24–27.
[94] M. Niazi, D. Wilson, D. Zowghi, A maturity model for the implementation of software process improvement: an empirical study, J. Syst. Softw. 74 (2) (2005) 155–172.
[95] A. Nugroho, M.R. Chaudron, A survey of the practice of design–code correspondence amongst professional software engineers, in: Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, IEEE, 2007, pp. 467–469.
[96] N. Panlilio-Yap, Software estimation using the slim tool, in: Proceedings of the 1992 Conference of the Centre for Advanced Studies on Collaborative Research-Volume 1, IBM Press, 1992, pp. 439–475.
[97] C. Park, G. Im, M. Keil, Overcoming the mum effect in it project reporting: Impacts of fault responsibility and time urgency, J. Assoc. Inf. Syst. 9 (7) (2008) 409.
[98] S. Paul, F. He, Time pressure, cultural diversity, psychological factors, and information sharing in short duration virtual teams, in: 2012 45th Hawaii International Conference on System Sciences, IEEE, 2012, pp. 149–158.
[99] D.J. Paulish, A.D. Carleton, Case studies of software-process-improvement measurement, Computer 27 (9) (1994) 50–57.
[100] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manage. Inf. Syst. 24 (3) (2007) 45–77.
[101] L.A. Perlow, Boundary control: the social ordering of work and family time in a high-tech corporation, Adm. Sci. Q. (1998) 328–357.
[102] L.A. Perlow, The time famine: toward a sociology of work time, Adm. Sci. Q. 44 (1) (1999) 57–81.
[103] L.A. Perlow, Time to coordinate: toward an understanding of work-time standards and norms in a multicountry study of software engineers, Work Occupations 28 (1) (2001) 91–111.
[104] L.A. Perlow, G.A. Okhuysen, N.P. Repenning, The speed trap: exploring the relationship between decision making and temporal context, Acad. Manage. J. 45 (5) (2002) 931–955.
[105] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering., in: Ease, vol. 8, 2008, pp. 68–77.
[106] D. Pfahl, An Integrated Approach to Simulation Based Learning in Support of Strategic and Project Management in Software Organisations, Fraunhofer-IRB-Verlag Stuttgart, Germany, 2001.
[107] L. Plonka, H. Sharp, J.v.d. Linden, Disengagement in pair programming: does it matter? in: Proceedings of the 34th International Conference on Software Engineering, IEEE Press, 2012, pp. 496–506.
[108] A. Powell, K. Mander, D. Brown, Strategies for lifecycle concurrency and iteration–a system dynamics approach, J. Syst. Softw. 46 (2–3) (1999) 151–161.
[109] M.L.N. PRICE Systems LLC, Your Guide to PRICE-S: Estimating Cost and Schedule of Software Development and Support, Technical Report, 1998.
[110] J.D. Procaccino, J.M. Verner, S.J. Lorenzet, Defining and contributing to software development success, Commun. ACM 49 (8) (2006) 79–83.
[111] L.H. Putnam, A general empirical solution to the macro software sizing and estimating problem, IEEE Trans. Softw. Eng. (4) (1978) 345–361.
[112] H. Rahmandad, D.M. Weiss, Dynamics of concurrent software development, Syst. Dyn. Rev. 25 (3) (2009) 224–249.
[113] S. Ramanujan, R.W. Scamell, J.R. Shah, An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort, J. Syst. Softw. 54 (2) (2000) 137–157.
[114] K. Reichelt, J. Lyneis, The dynamics of project performance: benchmarking the drivers of cost and schedule overrun, Eur. Manage. J. 17 (2) (1999) 135–150.
[115] N.O. Riordan, T. Acton, K. Conboy, W. Golden, It is about time: investigating the temporal parameters of decision-making in agile teams, in: Building Sustainable Information Systems, Springer, 2013, pp. 455–465.
[116] D. Rosenberg, B. Boehm, B. Wang, K. Qi, Rapid, evolutionary, reliable, scalable system and software development: the resilient agile process, in: Proceedings of the 2017 International Conference on Software and System Process, ACM, 2017, pp. 60–69.
[117] M. Ruiz, I. Ramos, M. Toro, A simplified model of software project dynamics, J. Syst. Softw. 59 (3) (2001) 299–309.
[118] I. Salman, B. Turhan, Effect of time-pressure on perceived and actual performance in functional software testing, in: Proceedings of the 2018 International Conference on Software and System Process, ACM, 2018, pp. 130–139.
[119] P.K. Sanjram, M. Gupta, Task difficulty and time constraint in programmer multitasking: an analysis of prospective memory performance and cognitive workload, Int. J. Green Comput. (IJGC) 4 (1) (2013) 35–57.
[120] S. Sawyer, R. Southwick, Temporal issues in information and communication technology-enabled organizational change: evidence from an enterprise systems implementation, Inf. Soc. 18 (4) (2002) 263–280.
[121] J. Schreier, The horrible world of video game crunch, 2016, [Online; posted 26-September-2016] https://kotaku.com/crunch-time-why-game-developers-work-such-insane-hours-1704744577.
[122] H. Shah, M.J. Harrold, S. Sinha, Global software testing under deadline pressure: vendor-side experiences, Inf. Softw. Technol. 56 (1) (2014) 6–19.
[123] D. Smite, C. Gencel, Why a CMMI level 5 company fails to meet the deadlines? in: International Conference on Product-Focused Software Process Improvement, Springer, 2009, pp. 87–95.
[124] R.K. Smith, J.E. Hale, A.S. Parrish, An empirical study using task assignment patterns to improve the accuracy of software effort estimation, IEEE Trans. Softw. Eng. 27 (3) (2001) 264–271.
[125] M. Sojer, O. Alexy, S. Kleinknecht, J. Henkel, Understanding the drivers of unethical programming behavior: the inappropriate reuse of internet-accessible code, J. Manage. Inf. Syst. 31 (3) (2014) 287–325.
[126] I. Sommerville, Software process models, ACM Comput. Surv. (CSUR) 28 (1) (1996) 269–271.
[127] B.C. Spilker, The effects of time pressure and knowledge on key word selection behavior in tax research, Account. Rev. (1995) 49–70.
[128] N. Staudenmayer, M. Tyre, L. Perlow, Time to change: temporal shifts as enablers of organizational change, Organ. Sci. 13 (5) (2002) 583–597, doi:10.1287/orsc.13.5.583.7813.
[129] A.J. Sutton, S. Duval, R. Tweedie, K.R. Abrams, D.R. Jones, Empirical assessment of effect of publication bias on meta-analyses, BMJ 320 (7249) (2000) 1574–1577.
[130] A. Tang, M.A. Babar, I. Gorton, J. Han, A survey of architecture design rationale, J. Syst. Softw. 79 (12) (2006) 1792–1804.
[131] A.H. Tapia, The power of myth in the IT workplace: creating a 24-hour workday during the dot-com bubble, Inf. Technol. People 17 (3) (2004) 303–326.
[132] C.-I. Teng, Y.-I.L. Shyu, W.-K. Chiou, H.-C. Fan, S.M. Lam, Interactive effects of nurse-experienced time pressure and burnout on patient safety: a cross-sectional survey, Int. J. Nurs. Stud. 47 (11) (2010) 1442–1450.
[133] H. Topi, J.S. Valacich, J.A. Hoffer, The effects of task complexity and time availability limitations on human performance in database query tasks, Int. J. Hum.-Comput. Stud. 62 (3) (2005) 349–379.
[134] S. Tuomivaara, H. Lindholm, M. Känsälä, Short-term physiological strain and recovery among employees working with agile and lean methods in software and embedded ICT systems, Int. J. Hum.-Comput.Interact. 33 (11) (2017) 857–867.
[135] R.T. Turley, J.M. Bieman, Competencies of exceptional and nonexceptional software engineers, J. Syst. Softw. 28 (1) (1995) 19–38.

[136] K.E. Van Oorschot, H. Akkermans, K. Sengupta, L. Van Wassenhove, Anatomy of a decision trap in complex new product development projects, Acad. Manage. J. 56 (1) (2013) 285–307.
[137] K.E. Van Oorschot, K. Sengupta, H. Akkermans, L. Van Wassenhove, Get fat fast: Surviving stage-gate® in NPD, J. Prod. Innovation Manage. 27 (6) (2010) 828–839.
[138] K.E. Van Oorschot, K. Sengupta, L. van Wassenhove, Dynamics of agile software development, in: Proceedings of the 27th International Conference of the System Dynamics Society, 2009.
[139] J. Verner, J. Sampson, N. Cerpa, What factors lead to software project failure? in: Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on, IEEE, 2008, pp. 71–80.
[140] M.J. Waller, J.M. Conte, C.B. Gibson, M.A. Carpenter, The effect of individual perceptions of deadlines on team performance, Acad. Manage. Rev. 26 (4) (2001) 586–600.
[141] E.V. Wilson, S. Sheetz, S. Djamasbi, J. Webber, Testing the group task demands-resources model among it professionals, in: 20th Americas Conference on Information Systems (AMCIS), 2014.
[142] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014, p. 38.
[143] D. Yang, Q. Wang, M. Li, Y. Yang, K. Ye, J. Du, A survey on software cost estimation in the chinese software industry, in: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, 2008, pp. 253–262.
[144] Y. Yang, Z. Chen, R. Valerdi, B. Boehm, Effect of schedule compression on project effort, in: Proceedings of the 5th Joint International Conference & Educational Workshop, the 15th Annual Conference for the Society of Cost Estimating and Analysis and the 27th Annual Conference of the International Society of Parametric Analysts, 2005.
[145] R.M. Yerkes, J.D. Dodson, The relation of strength of stimulus to rapidity of habit–formation, J. Comp. Neurol. 18 (5) (1908) 459–482.
[146] H. Zhang, M. Huo, B. Kitchenham, R. Jeffery, Qualitative simulation model for software engineering process, in: Software Engineering Conference, 2006. Australian, IEEE, 2006, pp. 10–pp.
[147] Y. Zhang, B.M. Wildemuth, Qualitative analysis of content, Applications of Social Research Methods to Questions in Information and Library Science, 318, 2016.
[148] W.W. Zung, A self-rating depression scale, Arch. Gen. Psychiatry 12 (1) (1965) 63–70.