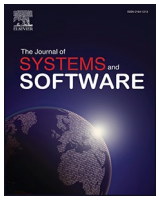
The Journal of Systems and Software 216 (2024) 112155

Contents lists available at ScienceDirect

The Journal of Systems & Software

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



A systematic literature review on characteristics of the front-end phase of agile software development projects and their connections to project success

Magne Jørgensen

Simula Metropolitan Center for Digital Engineering, Oslo, Norway

ARTICLE INFO

Keywords:
Agile software development
Cost estimation
Benefits estimation
Front-end phase

ABSTRACT

Context: Software development of new products and services often involves a front-end phase where user needs are analysed, costs and benefits are estimated, and initial plans are created.
Goal: This study aims to learn more about how the introduction of agile software development has affected practices and outcomes related to cost and benefit estimation in this front-end phase and to understand better what would improve this phase.
Method: We identified, reviewed and aggregated the results from 42 relevant research articles by searching literature databases and snowballing relevant articles.
Results: The front-end phase of agile was found to be, on average, similar and just as comprehensive as that of non-agile software development. This may be unfortunate, given the finding that more successful agile software development is connected with less detail in cost estimation and planning-related activities. A less comprehensive front-end phase may be especially beneficial for low-risk agile software development.
Conclusion: The results of this review suggest that agile principles, so far, have had a limited influence on the front-end phase. We recommend more flexibility and context-dependency in how the front-end phase of agile software development is conducted, including less comprehensive estimation and planning activities for low-risk software development contexts.

1. Introduction

The front-end phase of software development of products and services consists of proposing, evaluating, and selecting how to meet identified needs or opportunities and creating an initial plan for achieving this (Williams et al., 2019). This phase, which contains all activities preceding the execution of the actual software development, is well-documented to be essential, and both successes and failures can often be traced back to what was done or not done in this phase (Meier 2008, Edkins et al., 2013, Williams et al., 2019). Estimating the cost and benefits of the investments and providing the necessary information as input for the estimation process is an essential part of the front-end phase. Estimates of costs and benefits are, for example, central as input to deciding whether to invest, as input to funding or budget requests, and in creating the initial plan for software development. Names used as synonyms or overlapping with the front-end phase include "up-front phase", "pre-project phase", "project initiation phase", "initial phase", "project launching phase" and "sprint zero." An example of a stage-gate type of front-end phase (Larsen et al., 2021), mandatory for large governmental software development projects (in Norway), is displayed in Fig. 1 for illustrative purposes. Here, the front-end phase starts with the identification of needs. It then has a stage, termed "conceptual appraisal", where different alternative ways of meeting the identified needs are analyzed and compared. This stage includes a cost-benefit analysis of the alternatives and the selection of one of them. For the type of large governmental software development investments covered by this process, this part of the front-end phase is concluded by an external quality assurance (QA1). If the outcome of the quality assurance is positive and the national cabinet approves the software development investment, the project moves on to the next stage, the pre-project stage. In this stage, the estimates of cost, time and benefits are detailed and the plans for how to organize and execute the project are created. A final external quality assurance (QA2) is then completed, and if positive and the parliament approves the investment, the front-end phase is completed and the software development (the execution) can start.

The process in Fig. 1 is an example of a traditional and comprehensive front-end phase, meant for larger investments in software development and other construction activities. There are many less formal and less comprehensive variants of the front-end phase of

https://doi.org/10.1016/j.jss.2024.112155
Received 16 February 2024; Received in revised form 8 May 2024; Accepted 6 July 2024
Available online 8 July 2024

software development. Common for all of them is that they contain activities and deliveries executed *before* the actual software development starts. In particular, they include activities that motivate the investments, such as analyses of the feasibility of the proposed solutions and the profitability of the investment, and prepare for the execution of the software development, such as the development of plans for the execution of the project and the allocation of competent resources.

In the systematic literature review included in this paper, we focus on the estimates of costs and benefits provided in the front-end phase of agile software development and their connected processes. Such estimates are required for most software development initiatives regardless of the choice of the development method. However, the practices and comprehensiveness of the front-end phase may, and probably should, differ depending on the software development method. To address this, we contrast the front-end phases of agile and non-agile software development in our review.

The remaining paper is organised as follows: Section 2 briefly discusses possible implications of working agile for the front-end phase of software development. Section 3 motivates the need for our systematic literature review and presents the research questions.

Section 4 describes the review process. Section 5 presents the results. Section 6 discusses the results and their limitations. Section 7 concludes.

2. Agile software development and the front-end phase

In several types of non-agile[1] software development, such as stage-gate and waterfall-model inspired development methods, the goal of the front-end phase is often to identify and detail *all* requirements and estimates and do *all* the planning before the teams or projects are established and the actual development of the software starts (Cohen et al., 2003). Experience shows that this goal is hardly achievable, and requirement changes, re-design and re-planning are frequently needed (Rainer and Shepperd, 1999, Jørgensen, 2019). While this need for change in requirements, design and plans is acknowledged and supported in many non-agile software development models (Dima and Maassen, 2018), the underlying goal of completeness may remain and lead to comprehensive analysis and design, detailed requirements, cost estimates and plans. This, in turn, may lead to a long and demanding front-end phase. For example, (Blackburn et al., 1996) found the front-end phase of a non-agile software development project to require as much as 50% of the total elapsed time. To what extent and in which contexts comprehensive and detail-oriented front-end of non-agile software development lead to better outcomes has, as far as we are aware of and documented in our review, not been subject to much empirical research. Many, including (Boehm, 1996), have argued that the typical length and comprehensiveness of the front-end phase of software development may be problematic.

Agile software development was proposed, motivated by, among other things, the sometimes very long and perceived too detail-focused and comprehensive front-end phase of non-agile software development, along with other challenges (Cohen et al., 2003). The origins of this software development approach can be traced back to the evolutionary and iterative methodologies proposed in the 1950s and 1960s. Its widespread popularisation and adoption may, to some extent, be attributed to the establishment of the Agile Manifesto in 2001, as formulated at *agilemanifesto.org*, and its connected principles and practices.

Agile software development has become increasingly popular over the years, see (Hoda et al., 2018), and has several principles and practices potentially in conflict with how the front-end phase of software development typically has been completed. For example, agile software development aims at delivering value as early, as formulated in the principle, "*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*." This principle may conflict with the often long and comprehensive front-end phase of non-agile software development, making early deliveries hard. Agile software development also emphasises flexibility in what is developed, as stated in the principle: "*Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage*." This principle may conflict with putting much work into creating a detailed requirement specification in the front-end phase, which may not only be seen as a waste of time on things that are likely to, and perhaps even should, change. It may also limit the flexibility in providing the client with what gives the highest value. Finally, agile software development may not believe in detailed front-end phase work on how to build the software, and states in one of the principles that "*The best architectures, requirements, and designs emerge from self-organising teams*." For a more comprehensive discussion on conflicts, or tensions, between agile software development and non-agile front-end phases, see (Lappi and Aaltonen, 2017). For more on the various types of agile software development and how they have changed over the years, see (Dybå and Dingsøyr, 2008, Abrahamsson et al., 2017, Edison et al., 2021).

Based on the potential conflicts, one may expect that the front-end phase of agile software development would put less emphasis on the completeness and detail of the requirement specification, design, and architecture. We may also expect that the estimation of the cost and benefits of the deliverables is completed with less detail, only sufficient as input to a just-good-enough cost-benefits analysis and an initial, perhaps purposely, incomplete plan. Consequently, we may expect the typical front-end phase of agile software development to be shorter and require less effort than the typical front-end phase of non-agile software development. We may also expect a more frequent use of cost estimation methods typically connected with agile software development, such as relative estimation with the use of story points and group-based estimation based on "Planning Poker" (Cohn, 2005). For more on agile cost estimation methods see (Usman et al., 2014, Usman et al., 2015).

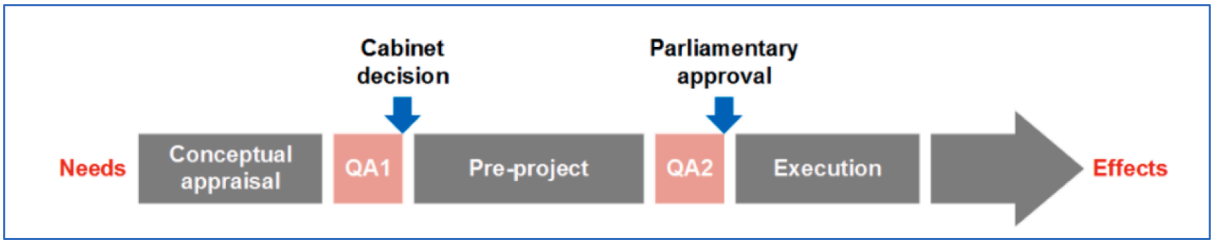


Fig. 1. Example of stages of the front-end phase of software development.

Agile *principles* may be nice to have, but what happens in practice in the front-end phase may be determined by other concerns. The main concerns of those investing in software development, such as the project or product owners, are probably not so much to what extent the software

[1] We use the terms "non-agile" for lack of a better term for the set of software practices that are not categorized as agile software development. Other labels put on non-agile methods are "traditional", "waterfall", "document-driven", and "plan-driven".

development is agile or not, but instead to what extent they can be confident that the investment is worthwhile, that the cost control will be good, that a project delivers on time, and that the expected benefits from the investment will be realised. It may consequently be that, in practice, much of the agile software development does not have a front-end phase following agile principles, but rather one similar to the front-end phase of more traditional, non-agile software development. Currently, we do not know much about this.

3. Motivation and research questions

The systematic literature review presented in this paper is motivated by the above concern, i.e., to what extent the front-end phases of agile and non-agile software development differ in practice and to what extent they should differ. The review focuses on estimating cost[2] and benefits completed in the front-end phase of agile software development, but also includes discussions on connected activities and processes.

Our review differs from prior literature reviews on estimation in agile software development, which has had a strong focus on cost estimation of agile software development conducted *after* the front-end phase has been completed, such as the reviews in (Usman et al., 2014, Kupiainen and Mäntylä, 2015, Bilgaiyan et al., 2017, Vyas et al., 2018, Fernández-Diego et al., 2020, Sudarmaningtyas and Mohamed, 2021). While estimation methods used during and after the front-end phase may overlap, there are essential differences between front-end phase estimation and estimation of agile software development completed during software development. The latter tend, for example, to use methods that require that development teams are established and historical data from previous work, such as "velocity" (productivity), is available (Mahnič and Hovelja, 2012). It also tends to focus on estimating the next sprint or iteration, not the whole investment or the total project. Front-end phase estimation focuses on estimating the cost and benefits of the whole investment or total project. It must often be done before knowing who will conduct the work and the teams are established. Our review also differs from prior reviews in that it only includes results from estimation in real-world contexts and that it includes results from the front-end phase estimation of benefits.

As documented in this review, there has been limited research interest in the front-end phase of agile software development. This lack of research focus on the front-end phase of agile software development may be paralleled by a similar lack of support for and discussion of the front-end phase cost and benefit estimation among agile software development practitioners. As an illustration, the seminal book on agile estimation (Cohn, 2005) does not cover front-end estimation of agile software development, just release, iteration, and task estimation.

Our review aims to answer the following research questions:

- Research Question 1: How are the costs and benefits estimated in the front-end of agile software development, how comprehensive are the front-end phase activities and how does the agile front-end phase differ from that of non-agile? Sub-questions include:
  - ○ What are the dominating estimation methods and estimation objects?
  - ○ How comprehensive are the estimation-related activities?
  - ○ What is the optimal level of detail and effort spent on the estimation-related work in the front-end phase?
- Research Question 2: How accurate are the front-end phase estimates of cost and benefits for agile software development, and how does the accuracy differ from that of non-agile software development?
- Research Question 3: What are the connections between estimation-related activities in the front-end phase and the outcome of agile software development? Do these reasons differ from those of non-agile software development?

By answering these questions, we hope to increase our knowledge about front-end estimation and recommend improved practices in the front-end phase of agile software development.

4. Review process

This section describes the inclusion and exclusion criteria for the literature review (Section 4.1), the search process (Section 4.2) and the review process (Section 4.3).

4.1. Inclusion and exclusion criteria

The criteria used for the *inclusion* of a research-based study are that the study reports:

- Empirical results from studies on estimating the cost or benefits of software development in real-world contexts.
- Results from estimation in contexts where the software development work is reported to be agile. This includes various variants and degrees of agile software development, e.g., so-called "hybrid" methods. It is sufficient for inclusion that at least some elements of the software development are described as agile, and there will consequently be many variants of agile software development included in the reviewed studies. If a data set includes data from other types of work than software development, it is only included if most (>50 %) observations are from software development contexts. If a study includes work processes characterised as "traditional", "stage-gate", "plan-driven", "waterfall", etc., i.e., non-agile software development, it is only included if there are separate analyses of agile and non-agile work processes, e.g., if some of the results compare the performance of agile and non-agile projects, or when the majority (>50 %) of the software development observations are characterised as agile.
- Results from estimation conducted in the front-end phase of software development. The front-end phase typically includes activities leading to developing cost and benefits estimates as input to a business case and the initial plan for executing the software development work. The software development work will typically be, but does not have to be, organised as a project.
- The estimated work is the agile software development of new products or services, not, for example, the further development or maintenance of existing software. We do, however, include estimation work for releases of existing software products when the releases require substantial work effort and involve developing new services.

The criteria used for the *exclusion* of a research-based study are that the study reports:

- Empirical results from studies on estimating the cost or benefits during the execution of agile software development, such as sprint or task estimation in software development teams. Studies that do not explicitly state the type of estimates they study are not included.
- Results using historical software development data to develop and/or evaluate formal estimation models or other estimation support, i.e., contexts where software professionals have not used the models or estimation support in a real-world work context.
- Results from studies using students as subjects.
- Results from studies simulating real-world behaviour.

[2] In software development the cost is typically dominated by the work cost, and the majority of the studies on cost estimation are for this reason studies on work-effort estimation. We will use effort and cost interchangeably in this paper.

In addition, we excluded non-peer-reviewed work, studies reported in languages other than English, and studies older than from the year 2000, which is about the time when agile software development started to get reported.

4.2. Search process

The search for relevant research studies followed the following steps:

i. Search in WebOfScience (June 2023) for articles from 2000 and younger using the search string: *((estimat* OR forecast* OR predict*) AND (cost OR effort OR benefit* OR value) AND (agil*))*.

Step i. resulted in 1.495 articles, which were reduced to 223 results when reducing the relevant topics ("*meso*") to "software engineering" and "management", and further reduced to 25 potentially relevant papers when reading through all titles and abstracts.

i. Repeated search in *scopus* (June 2023) with the same search string, limiting the search scope to the relevant document types ("ar" or "cp"), including only papers from year 2000 or younger, and limiting the scope to the subject areas "comp" and "engi".

Step ii. resulted in 6 new, potentially relevant papers, for a total of 31 (25+6) potentially relevant papers.

i. Forward and backward snowballing of the 31 potentially relevant papers.

Step iii. resulted in 35 new potentially relevant papers, for a total of 66 (31+35) potentially relevant papers.

i. Reading the full papers of the 66 potentially relevant papers.

Step iv. excluded 24 papers, leaving 42 that were evaluated to be consistent with the inclusion and exclusion criteria described in Section 4.1.

4.3. The review process

The review process started with repeated reading of all the papers to identify, extract, review and aggregate the following information:

- Year of publication
- Types of estimates examined
- Data collection method
- Number of observations
- Context of the observations
- Results related to Research Question 1, i.e., how the estimates of cost or benefits in the front-end phase of agile software development are produced, how comprehensive the front-end process is and how this differs from non-agile.
- Results related to Research Question 2, i.e., the accuracy of the estimates of cost or benefits in the front-end phase, and how this differs from non-agile.
- Results related to Research Question 3, i.e., the connections between estimation-related front-end characteristics and the outcome of the software development work and how this differs from non-agile.

The results reported in the reviewed papers are from empirical studies applying various data collection methods, number of observations, agile software development contexts, and quantitative and qualitative data analysis types. In our result aggregation process and summary of results, we emphasise studies reporting results based on actual measurements and studies with a higher number of observations, more than studies reporting on perceived relationships and fewer observations. We report individual results from each study, separated per research question, to enable the reader to form an independent summary of the results, potentially different from the one we present.

5. Results

This section contains the following subsections:

- Section 5.1 provides general information about the included papers and their relation to the research questions.
- Section 5.2 presents results on how cost and benefits are estimated in the front-end phase of agile software development, i.e., results related to Research Question 1 (RQ1).
- Section 5.3 presents results on the accuracy of cost and benefits produced in the front-end phase of agile software development, i.e., results related to Research Question 2 (RQ2).
- Section 5.4 presents results on characteristics of the front-end phase of agile software development connected with cost control and benefits realisation, i.e., results related to Research Question 3 (RQ3).

5.1. The included papers

Table 1 displays a few general characteristics of the 42 reviewed studies and their relevance to the three research questions. A few papers have multiple studies with multiple types of objects studied. The papers are sorted by year, with the most recent years at the top.

Aggregated, the 42 identified papers have the following characteristics, where one paper may belong to more than one category:

- 38 papers report on cost estimation, and 16 on benefits estimation.
- 27 papers report results related to RQ1, 26 to RQ2, and 17 to RQ3.
- 31 papers have software projects as their study object, 9 software organisations, 3 software professionals, and one software development teams.
- The median number of study objects was 60 for studies of projects, 3 for studies of organisations, 276 for studies of software professionals, and 185 for studies of teams.
- The papers analyse a total of 2673 projects, 30 organisations, 959 software professionals, and 185 teams.
- 22 papers used questionnaires, 22 used interviews, 17 used document analysis, and 2 used observation or action research as their data collection method.
- 27 of the papers were published in journals and 15 at conferences.
- The publications venues were:

  o Software engineering journals with 14 papers: Information and Software Technology (5), IEEE Transactions on Software Engineering (3), IET Software (2), Journal of Systems and Software (2), IEEE Software (1), Empirical Software Engineering (1).
  o Project management journals with 8 papers: Project Management Journal (3), International Journal of Information Systems and Project Management (2), International Journal of Project Management (1), International Journal of Managing Projects in Business (1), The Journal of Modern Project Management (1).
  o Journals from other disciplines with 5 papers: IEEE Access (1), European Journal of Information Systems (1), Journal of Business Research (1), Journal of Information and Organizational Sciences (1), and MIS Quarterly Executive (1).
  o Software development-related conferences with 11 of the papers: XP (2), ECIS (1), ESEM (1), Euromicro (1), IASTED SEA (1), ICE-CCME (1), IEEE GSE (1), IEEE SCCC (1), IEEE STC (1), ISD (1)
  o Management-oriented conferences with 3 papers: IEEE CBI (2), IRNOP (1).

Table 1
Papers included in the review.

Id	Estimate type studied	Data collection method and object studied	Context of study	Relevant research question(s)	Paper reference
B1	Cost, benefits	Document analysis and interviews, 25 projects	Military sector, worldwide	RQ2	Berg, H., & Ritschel, J. D. (2023). The characteristics of successful military IT projects: a cross-country empirical study. International Journal of Information Systems and Project Management, 11(2), 25–44.
B2	Cost	Document analysis, 308 projects	Military sector, USA	RQ1, RQ3	Long, D., Drylie, S., Ritschel, J. Y, & Koschnick, J. (2023). An Assessment of Rules of Thumb for Software Phase Management, and the Relationship between Phase Effort and Schedule Success. IEEE Transactions on Software Engineering, Early access article (December 2023).
B3	Benefits	Document analysis, interviews and questionnaires, 10 projects	Public sector, Norway	RQ1, RQ2, RQ3	Holgeid, K. K., Jørgensen, M., Volden, G. H., & Berg, H. (2023). Realising benefits in public IT projects: A multiple case study. IET Software, 17(1), 37–54.
B4	Benefits	Interviews, 9 projects	Public sector, Norway	RQ1, RQ3	Tanilkan, Sinan S., and Jo E. Hannay. (2022). "Perceived Challenges in Benefits Management-A Study of Public Sector Information Systems Engineering Projects." 2022 IEEE 24th Conference on Business Informatics (CBI). Vol. 1.
B5	Benefits	Interviews, 3 organisations	Organizations using a scaled agile framework	RQ1, RQ3	Marnewick, Carl, and Annlizé L. Marnewick. (2022) "Benefits realisation in an agile environment." International Journal of Project Management 40.4: 454–465.
B6	Cost	Document analysis and Interviews, 1 project	Public sector, Sweden	RQ2	Lindskog, Carin. (2022). "Tensions and ambidexterity: a case study of an agile project at a government agency." *International journal of information systems and project management* 10, no. 2: 5–23
B7	Cost	Document analysis and interviews, 30 projects	Public sector, Denmark	RQ2	Krancher, O., Madsen, C. Ø., Alami, A., & Petersson, C. (2022). Explanations for budget and schedule overrun revisited–a configurational perspective on IT projects. In Proceedings/ International Conference on Information Systems (ICIS).
B8	Cost, benefits	Questionnaires 109 projects	Banking sector, South-Africa	RQ1	Moloto, M., Harmse, A., & Zuva, T. (2021, October). Agile Methodology use factors that influence project performance in South African Banking sector-A case study. In 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) (pp. 1–6). IEEE.
B9	Cost	Document analysis and questionnaires, 635 software professionals and 185 teams	Dutch, banking sector company with teams worldwide	RQ1	Kula, E., Greuter, E., Van Deursen, A., & Gousios, G. (2021). Factors affecting on-time delivery in large-scale agile software development. IEEE Transactions on Software Engineering, 48 (9), 3573–3592.
B10	Cost, benefits	Questionnaires, 477 projects	Public and private sector, worldwide	RQ1, RQ2	Gemino, Andrew, Blaize Horner Reich, and Pedro M. Serrador. (2021). "Agile, traditional, and hybrid approaches to project success: is hybrid a poor second choice?." Project Management Journal 52.2: 161–175.
B11	Cost	Questionnaires, 155 projects	IT industry in Pakistan	RQ3	Aizaz, F., Khan, S. U. R., Khan, J. A., & Akhunzada, A. (2021). An empirical investigation of factors causing scope creep in agile global software development context: a conceptual model for project managers. IEEE Access, 9, 109166–109195.
B12	Cost, benefits	Questionnaires, 153 projects	Worldwide, mainly USA	RQ3	Butler, C. W., Vijayasarathy, L. R., & Roberts, N. (2020). Managing software development projects for success: Aligning plan-and agility-based approaches to project complexity and project dynamism. Project Management Journal, 51(3), 262–277.
B13	Cost	Document analysis, 1 project	Telecom	RQ2	Karna, H., Gotovac, S., Vicković, L., & Mihanović, L. (2020). The Effects of Turnover on Expert Effort Estimation. Journal of Information and Organizational Sciences, 44(1), 51–81.
B14	Cost, Benefits	Questionnaires, 147 projects	Public and private sector, Norway	RQ1, RQ2, RQ3	Holgeid, Knut Kjetil, and Magne Jørgensen. (2020). "Practices connected to perceived client benefits of software projects." IET Software 14.6: 677–683.
B15	Cost, benefits	Questionnaires, 71 projects	Public and private sector, Norway	RQ2, RQ3	Holgeid, Knut Kjetil, and Magne Jørgensen. (2020). "Benefits management and agile practices in software projects: how perceived benefits are impacted." 2020 IEEE 22nd Conference on Business Informatics (CBI). Vol. 2.
B16	Cost	Questionnaires, 276 software professionals	Internet product provider, worldwide	RQ1	Bianchi, M., Marzi, G., & Guerini, M. (2020). Agile, Stage-Gate and their combination: Exploring how they relate to performance in software development. Journal of Business Research, 110, 538–553
B17	Cost	Questionnaires, 102 projects	Private sector, mainly Israel	RQ2	Lishner, I., & Shtub, A. (2019). Measuring the success of Lean and Agile projects: Are cost, time, scope and quality equally important?. The Journal of Modern Project Management, 7(1).
B18	Cost, benefits	Questionnaires, 196 projects	Private and public sector, Norway	RQ1, RQ3	Jørgensen, M. (2019). Relationships between project size, agile practices, and successful software development: results and analysis. IEEE software, 36(2), 39–43.
B19	Cost	Interviews, 10 organisations	Private sector, Chile	RQ1, RQ2	Vera, Tomás, Sergio F. Ochoa, and Daniel Perovich. (2018). "Understanding the Software Development Effort Estimation in

(*continued on next page*)

Table 1 (*continued*)

Id	Estimate type studied	Data collection method and object studied	Context of study	Relevant research question(s)	Paper reference
					Chilean Small Enterprises." University of Chile, Santiago de Chile.
B20	Cost	Document analysis and interviews 60 projects from the same organisation	Telecom, Sweden	RQ1, RQ2	Usman, Muhammad, et al. (2018). "Effort estimation in large-scale software development: An industrial case study." Information and Software technology 99: 21–40.
B21	Cost	Document analysis, interviews and questionnaires, 1 organisation	Private sector, Brazil, Pakistan and Norway.	RQ1, RQ2, RQ3	Usman, M., Petersen, K., Börstler, J., & Neto, P. S. (2018). Developing and using checklists to improve software effort estimation: A multi-case study. Journal of Systems and Software, 146, 286–309.
B22	Cost, benefits	Questionnaires, 101 projects	Private and public sector, Norway	RQ2	Jørgensen, Magne. (2018). "Do agile methods work for large software projects?." Agile Processes in Software Engineering and Extreme Programming: 19th International Conference, XP 2018, Porto, Portugal, May 21–25, 2018, Proceedings 19. Springer International Publishing.
B23	Cost	Document analysis and interviews 1 organisation	Privat sector, international	RQ1, RQ2, RQ3	Bick, Saskia, et al. (2017). "Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings." IEEE Transactions on Software Engineering 44.10: 932–950.
B24	Cost	Questionnaires, 99 projects	Public and private sector, USA	RQ1	Raunak, M. S., & Binkley, D. (2017). Agile and other trends in software engineering. In 2017 IEEE 28th Annual Software Technology Conference (STC) (pp. 1–7).
B25	Cost, benefits	Document analysis and interviews 3 organisations	Public sector, Finland	RQ1	Lappi, Teemu, and Kirsi Aaltonen. (2017). "Project governance in public sector agile software projects." International Journal of Managing Projects in Business 10.2: 263–294.
B26	Cost	Document analysis, interviews and questionnaires, 26 projects from 2 organisations	Telecom, Netherlands	RQ1, RQ2	Huijgens, H., Van Deursen, A., & Van Solingen, R. (2017). The effects of perceived value and stakeholder satisfaction on software project impact. Information and Software Technology, 89, 19–36.
B27	Cost, benefits	Document analysis, interviews and questionnaires, 12 projects from 6 organisations, survey of 48 other projects	Public and private sector, worldwide	RQ1	Hobbs, Brian, and Yvan Petit. (2017). "Agile methods on large projects in large organizations." *Project Management Journal* 48.3: 3–19
B28	Cost	Questionnaires, 51 projects	Worldwide	RQ1, RQ2	Britto, R., Mendes, E., & Börstler, J. (2015). An empirical investigation on effort estimation in agile global software development. In 2015 IEEE 10th international conference on global software engineering (pp. 38–45). IEEE.
B29	Cost, benefits	Questionnaires, 127 projects	Public and private sector, Norway	RQ2, RQ3	Jørgensen, M. (2016). A survey on the characteristics of projects with success in delivering client benefits. Information and Software Technology, 78, 83–94.
B30	Cost	Interviews, 1 project	Private sector, Sweden	RQ3	Evbota, Felix, Eric Knauss, and Anna Sandberg. (2016j. "Scaling up the planning game: Collaboration challenges in large-scale agile product development." Agile Processes, in Software Engineering, and Extreme Programming: 17th International Conference, XP 2016, Edinburgh, UK, May 24–27, 2016, Proceedings 17. Springer International Publishing.
B31	Cost, benefits	Action research, 1 project	Public project, Spain	RQ1, RQ2, RQ3	Torrecilla-Salinas, Carlos Joaquín, et al. (2015). "Estimating, planning and managing Agile Web development projects under a value-based perspective." Information and Software Technology 61: 124–144.
B32	Cost	Document analysis and interviews, 2 projects from the same organisation	Telecom	RQ2	Tarhan, Ayca, and Seda Gunes Yilmaz. (2014) "Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process." *Information and software technology* 56.5: 477–494.
B33	Cost	Questionnaires, 115 projects	Worldwide	RQ1, RQ2	Nelson, R. R., & Morris, M. G. (2014). IT Project Estimation: Contemporary Practices and Management Guidelines. MIS Quarterly Executive, 13(1).
B34	Cost	Interviews, 4 organisations	Private sector, multinational organizations	RQ1, RQ2, RQ3	Lang, Michael, Kieran Conboy, and Siobhán Keaveney. (2013). "Cost estimation in agile software development projects." Information Systems Development: Reflections, Challenges and New Directions. Springer New York.
B35	Cost	Interviews, 3 projects from the same organisation	Private sector	RQ1	Daneva, Maya, et al. (2013). "Agile requirements prioritization in large-scale outsourced system projects: An empirical study." *Journal of systems and software* 86.5: 1333–1353
B36	Cost	Document analysis, interviews and observations, 12 projects	Internal and external projects	RQ1, RQ2	Cao, Lan, et al. (2013). "Adapting funding processes for agile IT projects: an empirical investigation." European Journal of Information Systems 22.2: 191–205.
B37	Cost	Document analysis and questionnaries, 192 projects	Worldwide	RQ2	Budzier, A., & Flyvbjerg, B. (2013). Making sense of the impact and importance of outliers in project management through the use of power laws. Proceedings of IRNOP (International Research Network on Organizing by Projects), At Oslo, 11.
B38	Cost	Interviews and questionnaires, 48 software professionals from 8 organisations, and 3 organisations	Private sector	RQ3	Magazinius, A., & Feldt, R. (2011). Confirming distortional behaviors in software cost estimation practice. In 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications (pp. 411–418). IEEE.

(*continued on next page*)

Table 1 (*continued*)

Id	Estimate type studied	Data collection method and object studied	Context of study	Relevant research question(s)	Paper reference
B39	Cost	Document analysis and interviews, 1 organisation	Telecom, Sweden	RQ2	Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study. Empirical Software Engineering, 15, 654–693.
B40	Cost	Questionnaires, 75 projects	Private sector	RQ1	Yang, Ye, et al. (2008). "Phase distribution of software development effort." Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement.
B41	Cost	Interviews, 4 organisations	Private sector	RQ1, RQ2, RQ3	Keaveney, Siobhan and Conboy, Kieran. (2006)."Cost estimation in agile development projects". ECIS 2006 Proceedings. 169.
B42	Benefits	Interviews, 2 projects from the same organisation	Private sector, Sweden	RQ1, RQ2	Dagnino, A., Smiley, K., Srikanth, H., Antón, A. I., & Williams, L. A. (2004, November). Experiences in applying agile software development practices in new product development. In IASTED Conf. on Software Engineering and Applications (Vol. 2004, pp. 501–506).

The number of publications per year is displayed in Fig. 2.

5.2. Front-end estimation practices

This section contains three sub-sections, all related to Research Question 1:

- The front-end cost estimation practices (Section 5.2.1)
- The front-end benefits estimation practices (Section 5.2.2)
- The comprehensiveness of the front-end phase (Section 5.2.3)
- The optimal level of detail and effort spent on the front-end phase (Section 5.2.4)

5.2.1. Cost estimation practices

Table 2 displays the reported practices on estimation objects and processes, including results comparing agile and non-agile. As in Table 1, the papers here and in the forthcoming result tables are sorted by year, with the most recent papers at the top.

As is evident from the findings in Table 2, there is no uniform way of reporting how cost estimation work is completed or what is used as estimation objects. In addition, the categories are not always precise in what they include. Using a "tool" for estimation seems, for example, to cover both formal estimation tools, such as COCOMO, and simple tools supporting expert estimation, such as a spreadsheet implementing a pre-made WBS with some supporting information. Similarly, analogy-based estimation seems to denote experts' use of analogies in a database or from memory, which belongs to expert estimation. Still, we cannot exclude that some estimation methods include using formal estimation models based on analogy.

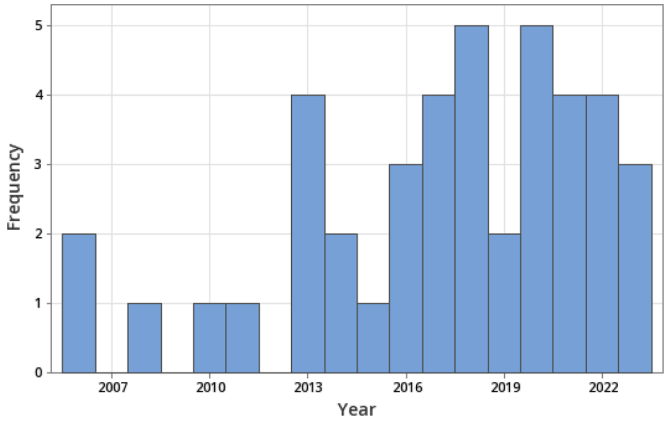


Fig. 2. Number of publications per year.

Even with these interpretation challenges, it is clear that nearly all of the reported estimation work in the front-end phase of agile software development is based on expert estimation and very little on truly formal estimation models. The type of expert estimation varies from the use of one person's expert judgment without any support from tools or databases of previous projects to more structured group-based estimation such as Planning Poker and expert judgment with support from historical data and WBS. In the few cases where formal estimation models have been reported used, they are either seldomly used, as in B24, or used together with other estimation methods, as in B28. An exception here is study B33, reporting that 20% had used a complex statistical formula. It seems, however, that this reflects that 20% of the respondents have used a complex statistical formula for estimation purposes at least once, perhaps in combination with expert judgment. This is different from claiming that many projects use formal estimation models regularly. Another exception is study B26, where all the projects studied in the two companies examined used function point-based estimation, which, according to the authors, exemplified the feasibility of using function points in agile contexts. However, the generality of this finding is limited by the fact that one of the selection criteria for the examined projects was that they could use function points, i.e., projects that could not use function points were excluded from the study. The use of function points requires quite a lot of up-front information about the requirements and the data structures, which means that the selected projects may not be representative of the front-end cost estimation of agile software development.

Most estimation work in the front-end phase of agile software development seems to have been based on estimating user stories or a collection of logically connected user stories (epics). However, a wide variety of other estimation objects were used, including the estimation of use cases, requirements, architectural elements, and decomposed work elements presented in a work breakdown structure (WBS).

The general picture, based on the comparisons of agile and non-agile estimation presented in Table 2, and comparing with results reported in prior reviews of non-agile software development, such as those in (Molokken and Jorgensen, 2003), is that the typical front-end phase cost estimation of agile software development is similar to that of non-agile software development and that both nearly always rely on expert

Table 2
Cost estimation objects and methods.

Paper	Main findings
B10	The most frequent estimation objects are use cases, user stories, and decomposed work elements from a Work-Breakdown Structure (WBS). Similar use of WBS and use cases, but more use of user stories when working agile compared to non-agile.
B19	The studied organisations used expert estimation. Similar use of expert estimation but less use of documented historical data when working agile compared to non-agile.
B20	Use of expert estimation. One important estimation object was the solution architecture. Separate estimates given for quotation and planning.
B21	Use of expert estimation. Estimation at the requirements level for large releases using story points.
B23	Estimation of epics using expert estimation.
B24	Most projects (near 40 %) were estimated using "expert opinion", typically based on a WBS. Other expert estimation techniques in use were "no formal technique" (21 %) and "story boards" (16 %). 11 % reported use of "function points" and 5 % had tool support, potentially including use of formal estimation models.
B25	Describes a non-agile front-end phase of agile software development, with similar estimation procedures and objects for agile as for non-agile front-end phases.
B26	The company used function point-based estimation based on user stories with extended information, such as information about "wireframes".
B27	The plans were based on the estimation of epics, user stories and high-level architecture.
B28	Expert estimation of various types: Planning poker (37 %), expert judgment (32 %), Analogy (13 %), Delphi (5 %). No use of formal estimation models as only method, but some use Use Case Points (8 %) and COCOMO (5 %) in combination with expert estimation.
B31	Expert estimation based on user stories. Use of relative estimation (story points), and estimation of velocity for the project.
B33	Expert estimation by comparing to similar past projects based on documented facts (43 %) or personal memory (57 %), by individuals reviewed in a group (57 %) or by an individual (37 %), by judgement without documented facts (67 %), by use of groups (70 %), by "guessing" (47 %). Elements of formality in the estimation methods included simple arithmetic formula (57 %), use of estimation tool (13 %), complex statistical formula (20 %). The objects estimated were user stories (using story points) (67 %), WBS (53 %), "work drivers" (33 %), function points (20 %) and lines of code (7 %). One project could use more than one estimation method. Similar estimation methods and objects for agile and non-agile, but less use of documented historical data, less use of WBS, and more use of story point and group-based estimation for agile.
B34	Estimates were given by use of expert estimation with various levels of formality and frequency in use of group-based estimation, typically based on user stories. No use of formal estimation models.
B35	Estimation of user stories extended with additional detail and information, leading to "delivery stories". No reported use of formal estimation models.
B36	Expert estimation of user stories and tasks.
B41	Expert estimation, which were either pure expert judgment or analogy-based estimation methods based on "tables", i.e., the use of categories of, for example, size and complexity with connected historical productivity data and expert judgement-based categorizations.

estimation.[3] There may, however, be a few differences, such as that front-end cost estimation of agile software development is more frequently based on estimation of user stories and group estimation, in particular Planning Poker, and less often supported by documented historical data.

The approaches used for the front-end estimation of agile software development seem similar to those used by the teams during the sprints or iterations of the software development, as documented in (Usman et al., 2015). On a more detailed level, however, the agile teams may have to use different methods, e.g., due to not knowing who will complete the work, less detailed knowledge about the work to be done and/or lack of productivity data (velocity measurement) from previous sprints or releases as input to the estimation process.

Based on the reviewed papers, we could not identify any trend or change in how cost estimation is completed in the front-end phase of agile software development over the years.

5.2.2. Benefits estimation

Benefits estimates are, amongst others, often needed to produce a business case, i.e., to support the analysis of whether an investment is worthwhile and, this way, to motivate funding of the software development. Despite this, not much is reported on how the benefits are estimated. We identified only one study, B31, describing a systematic method or approach to estimate the benefits. That study describes a method for estimating benefits based on estimating relative benefits through "value points", i.e., a method for benefits estimation similar to that of using story points for relative estimation of software development cost. The use of "value points"-based estimation seems to be similar to what has been proposed as "benefits points"-based estimation in (Hannay, 2021). An advantage of this type of relative estimation of benefits is that the benefits of all deliverables can be transferred to the same unit, i.e., all benefits have the unit "value points". As pointed out in B31, the alternative is that some benefits are given in monetary units, some in other quantitative units, and others with qualitative descriptions, which is challenging to compare as a basis for prioritisation. As with the relative cost estimation, estimates of relative benefits are, in practice, expert judgment-based. The practical experience reported with the type of benefits estimation proposed in B31 is good. Still, the experience is solely from one smaller project where those proposing the estimation method were those using it. More studies are needed to validate the method's usefulness.

The remaining studies in Table 3 provide indirect information about how benefits are estimated in agile software development only, mainly by reporting challenges and limitations.

The findings from the studies reported in Table 3 indicate that:

Table 3
Process elements and challenges of benefits estimation in the front-end phase.

Paper	Main findings
B3	There was a lack of a systematic and comprehensive benefits identification and estimation process and no standard interpretation of what was meant by "estimated benefits". Only three of the ten projects found it important to identify and estimate more than the benefits that could defend the business case. Most (66 %) of the benefits were not monetized, but either measured in non-monetary units or not quantified.
B4	It had been difficult to identify all benefits in the front-end phase, and to assign reliable numeric values to the identified benefits. Benefits identification and estimation was mainly used to motivate funding.
B5	Benefits identification was mainly used for motivation of the rationale of the project, i.e., to build the "business case". One of the organizations connected benefits to epics (which was later further divided into user story benefits) to support the prioritization based on the benefit-to-cost ratio during the project execution.
B14	Most projects (76 %) built business cases, which include quantified benefits. Only a few of the projects (31 %) have some elements of uncertainty and risk analysis connected with the benefits.
B25	All three organizations required business cases, with cost and benefits. One organization reported that they had no systematic method or tool for deriving the business case, and did not use the benefit estimates during the project execution.@@@@Another organization reported that they connected the benefits estimates with use cases. In general, the need to produce (comprehensive) business cases in the front-end phase, as part of the governance process for the projects, created significant tension, which impacted the performance of the projects.
B31	Reports good experience with the use of relative benefits estimation through "value points".

[3] Most of the literature reviews on the frequency use of estimation methods in agile and non-agile contexts seem to be based on the number of papers published about them, which means that they reflect the researchers' interest in models rather than the actual use of the methods in the industry. We believe this potentially leaves the false impression that cost estimation based on formal estimation models is frequently used. Here, we only refer to estimation methods in real-world software development contexts.

- In many cases, there are no use of well-defined or systematic processes for identifying and estimating the benefits that agile software development is meant to produce
- The benefits estimation is dominated by the need to produce a business case and ensure funding and less influenced by its potential use for prioritisation and benefits management during agile software development.
- The different types of benefits identified and estimated require a variety of evaluation scales, some monetary, some using other quantitative scales, and some non-quantitative. This implies a need for many approaches to estimate and evaluate them.
- Some organisations use the same units, such as epics, user stories, and use cases, for both cost and benefits estimation. This may, to some extent, be motivated by and enable a more straightforward prioritisation process of deliveries due to comparing the cost and benefits estimation of the same delivery units.

The most striking finding is the lack of studies on how benefits are estimated and used in the planning and management of agile software development. While benefits may be more diverse regarding types and how to estimate them compared to cost, this does not reduce the importance of estimating and evaluating them. Delivering benefits to someone is why we develop software, and more knowledge about how to identify, estimate, plan, follow up and realise them should, for this reason, be highly prioritised in practice and research. A focus on benefits management and realisation when working agile should, we believe, be especially central given the agile principle "*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*" This is, we believe, only to a limited extent reflected in the industrial practices reported in Table 2. Instead, the industrial and research focus may have been on the second part of the principle, i.e., the use of continuous delivery, on which there are numerous empirical studies (Laukkanen et al., 2017).

5.2.3. The comprehensiveness of the front-end phase

Table 4 presents results from studies on the comprehensiveness of front-end activities of agile software development, which includes activities giving input to the cost and benefits estimation, such as requirement specifications and overall solution design, cost and benefits estimation itself, and activities using the estimates, such as planning of the software development.

The studies reported in Table 4 report diverging results. While B2, B18, B19 and B40 suggest that about the same or more effort is spent or that the documents are more detailed in the front-end phase of agile compared to non-agile software development, B27, B35, B36, B41, and B42 report results consistent with that less effort is and less detail of documents are present for agile software development.

Examining the type of empirical methods and number of observations of the studies, as reported in Table 1, we see that the difference in study design potentially explains the difference in results. The studies based on larger sets of documented project data, such as B2, or questionnaire-based data from many respondents and many observations, such as B18 and B40, report that agile projects do not, on average, have a lighter front-end than non-agile projects. A heavier front-end phase of agile, with more effort spent on cost estimation, was also found in B19, with observations from ten different organisations. The opposite results are what is found in the interview or questionnaire-based studies, as in B27 and B35, where people were asked about the perceived effect of moving from non-agile to non-agile software development or about practices that are perceived to be time-saving compared to traditional front-end practices, as in B36 and B41. The study, with only two projects, i.e., B42, suffers from not discussing or controlling for factors other than size and complexity that may differ between the two projects.

As we judge the generality and strength of the results in Table 4, we interpret them as suggesting that the front-end phase of agile software development is, on average, approximately as comprehensive as that of non-agile. This does not exclude several occasions where the front-end phase of agile software development is less comprehensive, i.e., the perceived effects on the front-end phase from working agile are likely to reflect reality in several contexts.

Table 4
Comprehensiveness of the front-end activities.

Paper	Main findings
B2	The agile (military) software development projects spent about 25 %[i] of the total effort in the front-end phase, i.e., on activities related to requirement analysis, architecture, and design. This was reported as a "high" proportion compared to traditional projects.
B18	The detail in the upfront requirement specification and the project plan was reported as similar for agile and non-agile projects. 60 % of agile and 53 % of non-agile were perceived to have "low" or "very low" detail in the upfront project plan. 55 % of agile and 54 % of non-agile were perceived to have "low" or "very low" detail in the upfront requirement specification.
B19	Companies using agile methods spent more time performing the estimation than those using non-agile methods.
B27	Less perceived planning effort, on average only half the time, in current agile than in previous non-agile projects. On average only 34 % of the sprints were planned in the front-end phase. Several projects needed to plan all sprints in the front-end phase.
B35	Less perceived effort spent on detailing requirements up front in current agile projects compared to previous non-agile projects.
B36	Most of the projects did "just enough" front-end requirement gathering, architectural modeling and estimating to enable seed funding for the project to get it started.
B40	About the same proportion of effort (16% vs. 14 %) was spent on front-end planning and requirement-related activities for non-agile and agile (iterative) projects.
B41	Two of the four companies tried to keep the early estimates for their agile projects as high-level as possible and did not finalize the requirements work in the front-end phase.
B42	The agile project spent 10 % of the total effort and the non-agile project spent 20 % on front-end activities related to documenting requirements and planning. Projects comparable in size and complexity.

[i] The paper does not report the exact difference or the amount, but from the text, we argue that it can be derived that about 25 % of the effort was spent in the front-end phase on the agile (military) projects.

A further argument for believing in a, on average, similarity in the comprehensiveness of the front-end phase of agile and non-agile projects is the result reported from a large-scale questionnaire-based study (Serrador and Pinto, 2015). This study, which includes 1386 projects across multiple industries, was not included in this review due to a too high proportion (>50 %) of non-software development projects. That study reports that the proportion of effort spent on the front-end phase was independent of the project's level of agility.

A possible reason for agile and non-agile projects typically being similar in the front-end phase is pointed out in B25. That study reports that the front-end phase governance processes, i.e., the decisions of go/no-go and the funding/budgeting, do typically not separate between agile and non-agile software development, i.e., that agile and non-agile projects typically have the same requirements on detail and comprehensiveness from the project owners or sponsors. This situation, with the same non-agile front-end phase for agile work, was observed as creating a tension of conflict between the need for project control and predictability, desired by the project owner, and the agile characteristics of flexibility and "just enough" planning and specification work in the front-end phase, desired by the software developers, in the studies B6 and B36.

5.2.4. How comprehensive should the front-end phase be?

The most interesting question may be what the characteristics of the front-end phase of agile software development should look like for different contexts. This is a difficult question without much research, and we could only identify four studies addressing elements of this, see Table 5.

B2 suggests that there is an optimal proportion of effort usage on the

Table 5
Connections between front-end comprehensiveness and outcome.

Paper	Main findings
B2	Agile projects in the military sector, spending around 30 % of the total effort on front-end activities, here including the requirement, architecture, and design activities, had the least schedule overrun, i.e., there was an inverted U-shape of the connection between effort spent in the front-end and the schedule overrun. It is claimed that the smaller projects, those with less than 50.000 executable source lines of code, but not the larger projects, would have benefitted from more effort in the front-end phase in the studied context.
B8	Reduced upfront planning for agile projects was statistically significantly connected with better project performance, where performance was measured as a combination of benefits, cost, time, productivity, and quality performance.
B16	A more agile type of specification had a negative connection with cost accuracy, a positive connection with schedule accuracy, and a negative connection with quality. All connections were weak, and none were statistically significant. Agility in the specification was related to the level of detail, delay in detailing requirements, and openness for change.
B18	More detail in the front-end project plan was not connected with better performance for agile or for non-agile software projects. Less detail in the front-end requirements was weakly connected with better performance of agile, but not for non-agile software projects. More requirement flexibility (scope flexibility) was strongly connected to better performance for agile but worse performance for non-agile projects. Performance was measured as a combination of client benefits, cost control, time control, productivity, and technical quality.

front-end type of activities, i.e., that there can be both too much and too little work on front-end activities related to the estimation of cost and benefits. This "sweet spot" of the proportion of effort spent in the front-end phase for military agile software development projects is reported to be as high as 30 %. B2 also reports that, when comparing the observed and the optimal mean use of effort spent in the front-end phase, it was the smaller projects that deviated most and implicitly were likely to have benefitted most from a more comprehensive front-end phase. This may indicate the presence of the common belief that smaller agile projects need a lower proportion of effort spent in the front-end phase than larger ones, a belief not supported by the observations. Notice that neither cost control nor benefits realisation success, only schedule control, was measured in that study and that the context is military agile projects.

The other three studies, i.e., B8, B16, and B18, did not analyse *where* the optimal level of front-end phase work would be. Instead, they focused on to what extent a higher or lower level of planning and detail in requirement documents was connected with better performance. The results are mixed. B8 reports that reduced upfront planning improved the project performance for agile software development projects in the banking sector. B16 finds that less work on the initial requirement specification is weakly connected with less time overrun, more cost overrun and lower quality. B18 reports that the projects categorised, by the respondents themselves, as having a lower level of detail in the plans and higher flexibility in requirements, e.g., that not all requirements were of the type "must-have" and/or specified in very high detail, performed better when the software development was assessed as agile. The opposite, i.e., worse performance, was the case for non-agile projects with the same characteristics.

We interpret the above results, which are mixed, as indicating that, on average, more flexibility of requirements and plans in the front-end phase of agile projects typically is connected with better performance. The results are mixed, and none of the papers discuss whether a more comprehensive front-end phase is mainly an indicator of a more complex project, which is connected with less successful project execution, or whether it is causing worse project performance. There are, however, reasons to believe there may be a causal effect, as less detail in plans and more flexibility in requirements is more in adherence to an agile way of working, with continuous deliveries with feedback, learning and potentially re-prioritization of deliverables and requirement changes. This adherence is supported by other results in B18, reporting that a higher degree of requirement volatility was connected with better outcomes for agile software projects but not for non-agile software projects. Another possible reason for believing in causality is that more detail in the front-end phase deliverables is connected with an extended phase duration. The longer the front-end phase, the more likely the analyses and estimates will be outdated. In other words, the connection between a more comprehensive front-end phase and less project success for agile projects may not purely be correlational but may also have some casual elements.

In addition to the studies in Table 4, there are studies on agile software development projects reporting perceived challenges with too little effort spent on the specifications and plans in the front-end phase. The studies reported in B9, B20, B21, B23, B24, B28, and B34 found that incomplete, too general, or missing requirements were the causes of cost and time overruns in agile software projects. The studies reported in B30 and B36, on the other hand, report that less detail in planning was perceived to be essential to achieving the desired level of flexibility and good project performance. Interestingly, while the questionnaire part of the study in B9 reports problems related to incomplete requirement specification, the project document analysis of requirement refinements in the same paper finds no such connection. This may illustrate the context dependency of the optimal amount of work on the front-end phase. While there are instances experienced by the respondents where a missing or incomplete specification has led to project problems, it may still be the case that spending more effort on the requirement specification does not, on average, lead to fewer problems.

Due to the limited number of relevant studies on the optimal level of comprehensiveness in the context of agile software development, we examined results from four large-scale studies on the front-end phase of other types of projects. These papers were identified by the same review process as the included papers, but excluded from the ordinary review due to not sufficiently large proportion of agile software development projects. The main results of these studies were:

- The study reported in (Choo, 2014), including an analysis of 1558 projects in a computer manufacturing company, found an inverted U-shaped relationship between the proportion of the time spent on problem definition, i.e., a part of the front-end phase, and the project's success, where success was interpreted as shorter duration of the project. The optimal proportion of time spent on the problem definition was in this study found to be 20–30 % of the total time. It was also found that the relationship depended on project experience, where more experience gave lower duration and weaker inverted U-shape, and on project complexity, where higher complexity gave higher duration and sharper inverted U-shape on the connection between time spent on the front-end activity and the success.
- The study reported in (Serrador and Turner, 2015) first summarises the literature on the relationship between the amount of planning and project success. It identified 13 empirical studies claiming a positive effect, i.e., that more planning gives more success, one empirical study claiming no effect on success, and one empirical study claiming a negative relationship. The second part of the paper, based on an analysis of 1386 agile projects from various industries, reports an inverted U-shaped relationship between the percentage of effort spent on planning and the project success, with success measured as a combination of stakeholder, team, client, and user satisfaction. The study also reports an inverted U-shaped relationship between the percentage of effort spent on planning and the efficiency of the project, with efficiency measured as a combination of time control, budget control, and scope control. For both success measures, the proportion of the total effort that maximised the projects' success was 25 %, i.e., a percentage substantially higher than the mean proportion of 15 % reported by the respondents.
- The study in (Zwikael, 2020) reports from an analysis of 992 projects from various industrial domains, where 33 % are software projects. It finds that more planning was, on average, connected with better

project performance, measured as schedule and cost control. This connection was mainly the case for private-sector projects and not so much for public projects. For public projects, it was found that there was too much effort involved in planning for low-risk projects and too little effort for high-risk projects. The result suggests that the level of risk should determine the amount of effort spent in the front-end phase.

- The study in (Zwikael et al., 2021) overlaps, regarding project data, with (Zwikael, 2020) and analyses 2002 projects from software and other domains. It finds that more *strategic* planning, which includes activities related to making a project plan with deliverables, activities, and resource allocation, and more tactical planning, which includes activities aiming at detailing the plan, cost and time estimates, together with making plans for communication, risk, and procurement, was in general positively correlated with project efficiency, measured as schedule and cost control, and effectiveness, measured as customer evaluation of project performance and benefits. Using regression models, including both strategic and tactic types of planning, the risk of the projects, and the interaction between the types of planning and level of risk and control variables, they found a strong positive connection between more tactical planning and project efficiency for high-risk projects and a strong negative connection between more tactical planning and project efficiency for low-risk projects, i.e., more tactical planning was negative for low and positive for high-risk projects.

Summarising the limited and diverging results in this section is not easy. There are no doubt situations where too little effort spent in the front-end phase, e.g., too little detail and poorly understood requirements, has led to and will continue to lead to software development problems. On the other hand, it is likely that much detail in the requirements, estimates, and plans can harm the flexibility and performance of agile software development and increase the risk of outdated analyses. The reported inverted U-shape between the comprehensiveness of the front-end phase and the performance of agile software projects supports the reasonable claim that there is an optimal level of comprehensiveness of the front-end phase, i.e., that software development does not benefit infinitely from more effort spent on specifying and planning in the front-end phase. Exactly where the optimum is, is likely to depend on the context.

Very little is documented about the context-dependency of the optimal level of specification and planning. The above few studies indicate that the project size, as reported in B2, the level of agility, as reported in B18, and the level of risk (Zwikael 2020, Zwikael et al., 2021) are candidates for being included in risk-dependent guidelines for when to invest in a more comprehensive front-end phase, and when less. Agile software projects may, in general, benefit from less detail in both requirement specifications and plans than non-agile, smaller agile projects may need more effort in the front-end phase than what typically is spent, and lower-risk projects may need less front-end phase effort than higher-risk projects.

5.3. Accuracy of front-end phase estimates (RQ2)

This section examines the accuracy of front-end phase estimates of the cost (Section 5.3.1) and benefits (Section 5.3.2) of agile software development.

5.3.1. Front-end phase cost estimates

Table 6 displays the main findings related to the bias and accuracy of front-end phase cost and time (schedule) estimates of agile software development.

Table 6 shows a considerable variation in how cost and time estimation bias and accuracy of agile software development are reported. This makes a formal aggregation of accuracy and bias difficult. The studies show, however, a clear tendency towards underestimating the effort, budget, and time, a finding similar to previous findings from all types of software development contexts, see review in (Halkjelsvik and Jørgensen, 2012). The typical underestimation of the cost and time in agile software development is 10–40%, with a large variation from context to context.

Table 6
Estimation bias and accuracy of cost and time estimates.

Paper	Main findings
B1	The mean cost overrun was 10 % and the mean time overrun was 25 %. Non-agile projects had lower cost overrun (mean of 0.4 %), but higher time overrun (mean of 32 %).
B6	The project went under budget and was on schedule.
B7	More agility in a project led to better budget and schedule adherence, especially for important projects.
B10	There were less accurate estimates of budget and time for agile compared to traditional projects. The difference is not statistically significant.
B13	The project experienced a 4 % over-estimation of effort.
B17	The median time overrun was 10–40 % compared to the planned time. 18 % of the projects took more than 40 % longer than planned.
B19	Companies using agile software development produced more accurate effort estimates.
B20	The cost quotation estimates, i.e., the estimates provided early in the front-end phase and presented to the client to indicate the price, were too low and highly inaccurate. There was a median of 5 % underestimation and a median estimation error of 69 %. The planning estimates, i.e., the estimates used as input to the project plan, had even stronger underestimation, but lower estimation error. There was a median of 26 % underestimation and a median estimation error of 49 %.
B21	The organization had a median cost estimation bias toward underestimation of 27 % before introducing and only 5 % after introducing an estimation checklist.
B22	Most projects scored “acceptable” or better on both cost control (62 %) and time control (69 %). The most agile projects had the best cost and time control, especially for medium and large projects.
B23	The organization often had very imprecise effort estimates.
B26	The two organizations, which both used function points-based estimation processes, strongly overestimated the cost (median of 51 %) and underestimated the time (median of 34 %).
B28	There was a bias towards cost underestimation. The median value of the underestimation is in the range of 5–25 %. 7 % of the projects were underestimated with 50 % or more.
B29	Most projects (78 %) were “acceptable” or “successful” regarding budget control.
B31	The project had an effort underrun, but a time overrun.
B32	The agile project had 4 % better cost estimation accuracy than the similar non-agile project, mainly due to more accurate estimates of the test phase.
B33	Agile projects were less often “over budget” (33% vs. 47 %) and over schedule (20% vs. 33 %) than non-agile. Agile projects were more often “on budget” (33% vs. 20 %) and “on time” (33% vs. 20 %) than non-agile projects.
B34	There was a tendency towards cost underestimation.
B36	The project cost was underestimated by 100 %, due to new tasks arriving later in the development process.
B37	More agile projects had the same median (0% vs. 0 %) cost overrun, but less time overrun (3.7% vs 14.3 %) than less agile projects.
B39	Agile projects had more accurate cost estimates. The reason for this is claimed to be that the requirements were more stable in agile projects. In agile software development 96 % of the originally planned requirements were implemented, while this was only the case and for 74 % of the requirements for non-agile projects.
B41	Typically, the cost estimates were accurate and within +/- 10 %,

The majority of the studies, i.e., B7, B19, B22, B29, B32, B33, B37, and B39, comparing the estimation accuracy or cost control of agile and non-agile software reports that the estimates of agile software development were less biased and more accurate than those of non-agile software development. Only two studies, i.e., B1 and B10, report the opposite. B1 reported a mean cost overrun of military non-agile software development projects close to zero. In contrast, the corresponding value for agile projects was 10 %, i.e., the estimates in both contexts were rater accurate. The same study reports that the time deviation was slightly lower for the agile than non-agile projects, with 25% vs. 32 % overruns, respectively. In B10, the difference between agile and non-agile was not statistically significant and based on an indirect, regression model-

based, method of measuring the impact of the development method on estimation accuracy, which may limit the robustness of the result. The totality of the evidence is consequently in favor of that front-end phase cost estimates of agile software development are, on average, more accurate than those of non-agile. This result is consistent with results found in other project domains. The study reported in (Nguyen and Mohamed, 2020), including 136 projects of various types, reports that the internal "agility-ability-to-react-to-change" had a positive effect on budget performance, especially for low-complexity projects.

5.3.2. Front-end phase benefits estimates

The studies report several challenges connected with evaluating the bias and accuracy of estimated benefits. The challenges include that many of the realised benefits are difficult to measure and compare with the estimated benefits (B4), that the organization do not even measure the benefits (B5), and that the realisation of benefits is not completed before some time after completion of the software development (B3). These challenges may explain why there are not many papers evaluating the actual realisation of the benefits. Table 7 summarises the sparse evidence we identified with results on the accuracy of front-end phase benefits estimates of agile software development, results on the success in realising the estimated or planned benefits, and results comparing benefits delivered by agile and non-agile software development.

As indicated by the sparse evidence, estimating and evaluating benefits seem to be a less mature field than estimating and evaluating costs. There were, for example, no studies that reported on the numerical deviation between the estimated and the realised benefits. Instead, the evaluation was more indirect, e.g., through the proportion of planned benefits that had been realised, or the client satisfaction with the software, assuming that this correlates with how much of the total benefits that had been realised.

We see from Table 7 that the two most recent studies, i.e., B1 and B3, report good benefits realisation in agile software development, with the projects expecting to deliver around 90 % of the planned benefits. Other studies, i.e., B14, B22 and B29, which use more perception-based evaluation measures, present results consistent with the above, with around 90 % of projects being categorised as acceptable or better regarding delivered benefits.

The studies comparing agile and non-agile regarding delivered client benefits, i.e., B1, B10, B14, B15, B29 and B42, all reported better benefits realisation, and implicitly better correspondence between planned and realised benefits, with the use of agile rather than non-agile software development.

Table 7
Estimation bias and accuracy of benefits estimates.

Paper	Main findings
B1	86 % of the planned benefits were realized for the agile projects. This was higher than the 72 % of benefits realized for non-agile projects.
B3	One year after project termination on average 45 % of the planned benefits were realized. There was a large realization variation between the projects with percentages varying from 13 % to 92 %. When asked to assess how much of the planned benefits would be realized in the future, the average increased to 92 %.
B6	The project delivered significantly more than was originally intended.
B10	Both agile and "hybrid" software development, with similar coefficients in a regression model explaining stakeholder success, had a better impact on benefits than non-agile ("traditional") software development.
B14	35 % of the projects delivered very good client benefits, 36 % good benefits, 21 % acceptable benefits and 4 % low/very low benefits. Agile projects were more successful in realizing client benefits.
B15	The median benefit score for agile projects was statistically significantly higher than for non-agile projects, with a score of 2 for agile and 1 for non-agile. Scale from −2 (very low benefits realization) to 2 (very high benefits realization).
B22	90 % of the projects were categorized as acceptable (55 %) or successful (35 %) regarding realized client benefits.
B29	95 % of the projects were categorized as acceptable (59 %) or successful (36 %) regarding realized client benefits.
B42	Higher client satisfaction for the agile than for the non-agile project. The two analyzed projects were comparable in size and complexity.

5.4. Connections between front-end phase and outcome

This section covers reported connections between elements of the front-end phase and the outcome of agile software development. We only report elements that are part of front-end phase processes. This excludes elements that belong to the project itself and elements belonging to the execution of the software development. Table 8 reports relevant results.

Central findings related to characteristics of the front-end cost and time estimation process connected with better outcomes reported include more flexibility in the scope of the projects, the use of a company-tailored checklist for the cost estimation work, the use of relative estimation of cost, and the use of experimentation through the use of "spikes" to get better knowledge before estimating the cost.

Table 8
Reported connections between front-end estimation practice and outcome.

Paper	Main findings
B3	A higher proportion of realised benefits was connected with well-formulated benefits. The degree of well-formulated benefits was assessed using the SMARC evaluation framework proposed by the authors, evaluating to which degree the benefits were Specified, Measurable, Accountable, Realistic and Comprehensive. In particular, the measurability (evaluability) of the benefits was connected with better benefits realization success.
B4	Good benefits realisation was connected with benefits that were well-connected to the specified tasks (deliverables).
B5	Project success was connected with well-defined benefits (targets).
B12	For agile projects, a flexible scope was connected with more project success, while for non-agile projects this led to less project success. Success was defined as a combination of cost and time accuracy and other success factors.
B11	Managers' overconfidence, budget constraints, lack of knowledge, unclear goals, and unrealistic expectations was connected with less cost control.
B14	Characteristics connected with more success in realizing benefits were scope flexibility, and the presence of a plan for how to realise and evaluate the benefits.
B15	The presence of a benefits management plan and planned evaluation of realized benefits were characteristics connected with more success in realizing benefits. Reasons for benefits shortfall included over-optimism in the estimation of the benefits, poor estimation process and deliberate over-estimation of the benefits.
B18	High scope flexibility was connected with more success of agile, but not for non-agile projects. Success is defined as a combination of cost and time accuracy and other success factors.
B21	The use of a company-tailored estimation checklist improved the effort estimation accuracy, even for the very experienced estimators.
B23	Effort estimation accuracy was negatively impacted by the lack of knowledge of those estimating, especially regarding lack of knowledge about inter-team dependencies.
B29	Characteristics connected with good benefits realization included having a benefits management plan, and contracts based on payment per hour rather than fixed price
B30	More accurate cost estimates were connected with plans that focused less on releases and more on continuous deployment and flexibility. Unclear requirements, which were connected with a lack of knowledge about the underlying user needs of features, were connected with inaccurate cost estimates.
B31	Relative cost estimates were easier to re-estimate, which in turn gave better project cost control.
B34	Accurate cost estimation was challenged by management and client pressure and users' lack of understanding of their own requirements.
B38	Cost estimates in agile projects were, compared to non-agile projects, more often affected by personal and organizational agendas, but less often by increase/decrease of estimates prior to negotiations. Agile and non-agile projects were similarly affected by inadequate communication, overlooked tasks and risk and increase in estimate to avoid overspending.
B41	Reasons for inaccurate cost estimates included poor communication, lack of technical expertise of those estimating, lack of formal estimation process, and lack of use of historical data. Use of "spikes", i.e., the implementation of a part of the deliverables, to get knowledge before estimating enabled more accurate cost estimates.

Characteristics of the front-end cost estimation process connected with worse outcomes include the use of inexperienced or less knowledgeable people in the estimation work, management or client pressure on lowering the cost estimates, poor communication, lack of a formal estimation process and lack of use of historical data. Most of these cost estimation elements have also been reported as important for the success of non-agile software development projects (Furulund and Moløkken-Østvold, 2007), and projects from other disciplines, see for example, (Williams et al., 2019). This suggests a similarity between agile and non-agile for many of the connections between the front-end phase elements and the outcome. A notable exception may be that increasing scope flexibility in the planning seems connected with better outcomes for agile but not for non-agile.

The findings related to characteristics of the front-end process connected with better benefits realisation emphasise the importance of proper formulation of the benefits, which should be formulated so that their realisation can be evaluated and be well-connected with deliverables for prioritisation during the execution of the software development. There should also be a plan for how to realise the benefits. As for cost estimation, planning for flexibility, such as having a flexible scope and using contracts based on hourly payment rather than a fixed price, is connected with better outcomes. Reasons for the benefits shortfall were related over-optimism, poor estimation process and deliberate overestimation of the estimate.

6. Discussion and limitations

This section discusses and summarises the results. Section 6.1 reflects on the research interest in the front-end phase of agile software development. Sections 6.2–6.4 discuss and summarise results related to Research Question 1, Section 6.5 discusses Research Question 2, Section 6.6 discusses Research Question 3, and Section 6.7 discusses some of the limitations of our review.

6.1. The research interest in the front-end phase of agile software development

Even the most agile software development typically requires an initial cost-benefit analysis of needs, alternative ways of solving the needs motivating the software development, cost and benefits of investing in the software development and, of worthwhile the investment, an initial plan before starting the actual software development work. These activities, which form what we term the front-end phase in this paper, will vary in what is done and how much is done from context to context. The front-end phase may even include programming, such as experimentation or "spikes", to evaluate or better understand alternative ways of solving the needs. The larger the investment, the larger the wish of those investing in software development to know whether the cost-benefit is positive and whether there is a realistic plan for completing the software development.

Despite the apparent importance, the research interest in the front-end phase of agile software development has been low. This is illustrated by the fact that very few of the 42 papers identified and reviewed in this paper had their focus on the front-end phase. Instead, the front-end relevant results were often reported as side results or contextual information of the main study focus. This is a situation quite different from that of other production fields. The review presented in (Williams et al., 2019) found, for example, 524 papers with a particular focus on the front-end phase, even after excluding papers not published in high-quality journals.

One possible reason for the low research interest in the front-end phase of agile software development may, we believe, be a perception that being agile, you should hardly have a front-end phase at all but instead start developing software as soon as practically possible. In other words, a research focus on the front-end phase of agile software development would focus on something that hardly should be there. This reason is supported by the fact that among the few papers focusing on the front-end of agile software development, the focus is often, see for example, B6 and B27, on the tensions and conflicts between front-end activities, such as producing requirement specifications and plans and working agile. Another possible reason for the low research interest may be that the most exciting parts of agile software development are considered to be what happens in the software development teams during the ongoing, perhaps product rather than project-oriented, software development work. This is consistent with the observation that the agile manifesto and principles only indirectly, or not at all, address the front-end phase. It is also consistent with the lack of attention to cost estimation approaches applicable in the front-end phase in agile communities. In any case, we believe there is a need for more research on the front-end phase, which establishes the foundation for the remaining work for both agile and non-agile software development.

6.2. The similarity of the front-end phase of agile and non-agile

As far as we know, the results from the literature review in this paper are the first attempt to summarise what we know about what goes on in the front-end phase of agile software development. Perhaps surprisingly to many, the typical front-end of agile software development is quite similar, and not less comprehensive, to that of non-agile software development.

This does not mean there should be no differences in the front-end phase of agile and non-agile software development. Among the reviewed studies, there are indications that agile software development may benefit from less detail in the estimation and planning activities than non-agile software development. Less detail may give more flexibility and opportunities for good agile practices in the execution phase and, consequently, better outcomes.

Neither does the above result mean that there are no examples of agile software development with substantially more lightweight front-end phases than traditionally has been the case for non-agile software development. Our review has identified several such examples.

The similarities between the front-end phases of agile and non-agile software development do, however, suggest that even after more than twenty years of agile software development, the agile principles have not affected the front-end phase much. Furthermore, there are no clear time-related patterns that indicate that this is about to change. We did, for example, not observe that more software development projects were less comprehensive and more in adherence to agile practices in their front-end phase over time. This may, to some extent, be because those investing in the software do not separate between agile and non-agile software when requiring information to support the investment decision and the need for proper planning. It may also be due to a low interest in the front-end phase by the agile communities. Agile teams, which in practice may be the typical focus of agile software development practices, have often not yet been formed at the time of executing the front-end phase.

An interesting observation, at least from a researcher's viewpoint, is that if we had emphasised the results from case studies interviewing people about what they had experienced or perceived happened with the front-end activities when moving from non-agile to agile software development, instead of relying more on the larger scale studies collecting less perception-based information about the front-end phase, our conclusion would probably have been different. In the experience- and perception-based studies, the respondents typically answered that they got a lighter front-end phase when moving to more agile software development. How to explain this difference in results dependent on the data collection method is not apparent. It may be caused by how the cases for interview-based studies on the effect of introducing agile are selected, e.g., mainly those companies succeeding in implementing a more agile front-end have been selected as exciting cases. It may also point at challenges faced by the respondents in separating what should have been the case when introducing agile software development and

what actually has been the case.

6.3. Estimation of cost and benefits

There were no truly surprising findings regarding how cost and benefits were estimated in the front-end phase of agile software development. As reported in earlier studies on non-agile software development, the by far dominant methods for estimation of costs and benefits were variants of expert estimation, i.e., estimates based on expert judgment by individual experts or experts in groups. The estimated units varied a lot, both for the estimation of cost and benefits. For cost estimation, the variation covers differences in types of cost elements, e.g., the estimation of user stories or requirements in a work breakdown structure (WBS), and the level of detail of these elements, e.g., the estimation at the level of epics or user stories.

Regarding the estimation of benefits, the results suggest that the estimation methods were selected based on the type of benefits to be estimated, sometimes in a rather ad-hoc manner.

As we interpret the results, the front-end cost and benefits estimation challenges and methods seem, to a large extent, not to have changed much due to implementing agile software development. If anything, and based on limited evidence, there has been a change towards more frequent use of user stories as cost estimation objects and group estimation as estimation methods, and perhaps less use of historical data with more use of agile software development.

6.4. The optimal comprehensiveness of the front-end phase

There were few studies addressing how to conduct the front-end phase of agile software development, including how comprehensive this phase should be. One possible reason for this, as illustrated by mixed results in the reviewed papers, is the high number of contextual factors that may determine the optimal detail and completeness of analyses and deliverables in the front-end phase.

A result derived from one study of the front-end of agile software development and two studies of the front-end of other types of agile projects suggests an inverted U-shape connecting the degree of comprehensiveness of the front-end phase and successful completion of work. This means that there seems to be a point where more detail and more analysis in the front-end phase leads to worse, not better, outcomes. The studies reporting on this suggest that the optimal point, on average, is when between 20 and 30 % of the total time and effort is spent on the front-end phase. Although we see no reason for doubting that there is an optimal point for a given context, there are several challenges with the reported results on where that optimum is. The results from the agile software development are from the military sector and the others from non-software development sectors. The optimal point of 20–30 % may consequently not represent most agile software development. Another challenge is the success measures used in these papers, which do not report on the perhaps most essential success elements of software development, such as delivered benefits and achieved productivity, but instead, only successes related to on time, on cost and with the specified functionality. Another challenge is that the optimal point is likely to depend on many contextual factors, i.e., knowing the average optimum for a set of projects, including a variety of contexts, may be of limited usefulness.

Despite these challenges, we believe there are relevant results with potential implications for practice. This is, in particular, the case for the finding that agile software development with less detailed upfront planning, on average, did better. Although this finding was not very strong, and one study found the opposite, the finding is consistent with the benefits of a more lightweight front-end phase for agile software development than what is currently typically the practice. As suggested in several of the reviewed papers, the main underlying mechanism for a benefit from fewer details may be that it increases the flexibility in the execution, which is connected with better opportunities to succeed with good agile practices and benefits management.

Our review identified studies in which insufficient comprehensiveness, leading, for example, to misunderstood or missing requirements of the front-end phase, was reported as causing problems with agile software development. This means that while agile software development, on average, may benefit from less detail in specifications and plans, the same lack of comprehensiveness may also, in some software development contexts, be the cause of problems.

Few papers examined when to have a more or less comprehensive front-end phase. Those that did, mainly from non-software, agile contexts, suggest that lower risk, which is connected with lower complexity and more experience with similar software development of work, is connected with benefits from having a lower level of comprehensiveness of the front-end phase.

6.5. The accuracy of the estimates of cost and benefits

The differences in how the accuracy and bias of front-end cost and time estimates were reported in the studies make result aggregation difficult. However, both the bias and the accuracy of the front-end cost and time estimates typically were 10–40 % and had a clear bias towards cost and time overrun. This suggests a similar cost and time estimation accuracy and bias as reported in previous studies of non-agile software development. When only considering the studies that compare the accuracy of bias of agile and non-agile software development, we find that most studies report that cost and time estimates of agile software development are more accurate and less biased than non-agile. This result of better accuracy and less bias of cost estimates for agile software development is similar to the results reported for agile projects in other production domains.

None of the studies reports on reasons for better accuracy and less bias connected to the choice of development methods. Therefore, it is hard to say whether better accuracy and less bias are caused by more flexibility in what is delivered in agile contexts, that agile software development results in processes with better cost control, or other reasons. The results suggest, nevertheless, that the fear that agile software development should lead to less controllable, perhaps even chaotic, software development is far from what seems to have happened (Wang and Vidgen, 2007).

The ways accuracy and bias of estimates of the benefits are reported are even more varied than for cost and time estimates. This may result from the fact that benefits are of many types, including monetary benefits, time savings, and other quantitative and qualitative benefits. In addition, it is often challenging to know to what extent the benefits actually have been or will be realised. This challenge is partly due to problems of how to measure the realised benefits and partly because much of the benefits will be realised after the software development work has finished. Despite these challenges, we identified a few studies that reported on agile software development's success in realising benefits. The limited findings suggest that, on average, agile software development is successful in realising benefits. Even more so, the studies comparing agile and non-agile software development reported better benefits realisation and implicitly better correspondence between planned and realised benefits using agile rather than non-agile software development.

6.6. Connections between elements of the front-end phase and the outcome

What is done in the front-end phase may influence the success of the software development. Front-end elements reported to be connected with better outcomes of agile software development include planning for flexibility, the use of a company-tailored checklist for the cost estimation work, the use of relative cost estimation, the use of experimentation in the front-end phase, proper formulation of the benefits, making plans for how to realise the benefits, and contracts based on hourly

payment rather than a fixed price. The front-end elements connected with worse outcome of agile software development include the use of inexperienced or less knowledgeable people in the estimation work, pressure to lower the cost estimates or increase the benefits estimates, overoptimism, poor communication, lack of a formal estimation process, and lack of use of historical data.

A front-end estimation element one might have expected to be connected with better or worse outcomes is the quality of the business case, i.e., the quality of the cost-benefit analysis motivating the investment in software development. Instead, this element was *not* connected with better or worse realisation of benefits, as reported in B3, B15 and B29, a finding consistent with that of a large-scale study of 378 projects of various non-software development types (Bechtel et al., 2021). This lack of connection should probably not be interpreted as indicating that business cases are unimportant, but perhaps instead that the business case is frequently developed mainly for receiving funding and being allowed to start the software development and not much used as a means for better benefits management, see for example B3. It is, however, also possible, as claimed in (Bechtel et al., 2021), that the potential positive effects of producing business cases, in several cases, may be neutralised by the limitations it puts on the freedom and creativity of agile teams. This illustrates, we think, the complexities in detecting how much and in which ways different factors in the front-end phase of agile software development affect the outcome. It also illustrates how little we know about these connections. To gain more robust and useful knowledge, we may need research studies that include more of the software development context and aim at understanding the causal mechanisms rather than just simply calculating the average connection.

6.7. Limitations

Essential limitations of the review reported in this paper include:

- What is and what should be termed agile software development is not always clear, and a set of well-defined, authoritative criteria to decide whether something qualifies as agile or not does not exist. In addition, agile practices and what is considered agile may have changed over time. This means that what is presented as agile software development in this review will likely cover a large variation of agile practices and degrees of agility. Some studies reduce this problem by using an agility scale, where, for example, the agility level depends on the number of agile practices implemented. Unfortunately, such studies are few, and most treat agile as a binary construct. The comparison of agile and non-agile front-end phase characteristics should consequently be interpreted with care and only as comparing larger groups of what typically is perceived as agile and as non-agile software development.
- It is not always clear what is meant by front-end phase estimates of cost or benefits, and the meaning may differ depending on the purpose, e.g., whether the estimates are given as input to the cost-benefit analysis or the initial plan. This lack of precision, e.g., whether a cost estimate is meant to be the best-case cost, most likely cost, expected (mean) cost, or a risk-averse (unlikely to exceed) estimate to be used for budget purposes, affects, for example, our ability to give a context-sensitive accurate summary of the accuracy of estimates of cost and benefits of agile software development. This lack of clarity in what is meant by an "estimate" is typical for most studies on cost estimation, see (Jørgensen, 2014), and we believe it is fair to assume that the estimation results in our review are comparable with those in previous studies and between agile and non-agile contexts.
- The representativeness of the studied contexts and the research results is often hard to judge. This may, in particular, be a challenge for studies of single cases, where for example, an agile project is studied because of its large size or successful implementation, or an organisation is studied because of its successful introduction of agile development. In addition, it is unclear to what extent findings from many years back represent today's agile software development or to what extent results dominated by one country's agile projects can be transferred to other countries. Fortunately, some of the included studies analyse larger sets of software projects and organizations from many countries. When results from single cases or narrow populations agree with results from larger-scale international studies, there are more reasons to believe in their validity and generality.
- There are different levels of subjectivity of the collected data, from pure experience-based judgments (perceptions) to statistical analyses on measured data from a large number of observations. While all variants of data collection may result in valid data, they may point to different types of relationships. For example, an experienced (perceived) connection between too little detail in the requirement specifications and cost overruns may be strongly influenced by recalling a few occasions where misunderstood requirements led to re-work and contributed to cost overruns and much less on the perhaps less noticeable occasions where the lack of detail in requirement specification increased the flexibility and had a positive impact on estimation accuracy, see our paper on what typically determine the perceived importance in questionnaire-based studies (Jørgensen, 2023). Statistical analyses on a larger set of observations may, on the other hand, find that there, on average, is no statistically significant connection between detail in requirement specification and cost overrun, implicitly ignoring a few cases where such connections were present. A summary of results from studies using different data collection methods is not straightforward and may, for example, need to separate between the typical (average) case and cases documenting the variance in observations. Optimally, the data should enable us to establish the conditions or contexts for when a connection exists and when not. This limitation should not be interpreted as that we know nothing about the influence of the contexts, just that there are only a few context effects that have been analyzed and need for more research.
- Only a few of the reviewed papers had their main research focus on front-end phase estimation of the cost or benefits of agile software development. This means that we could not expect all papers to report comprehensively on how the estimation was done or how the estimation work is connected with the success of the software development. It also means that some of the relevant information had to be derived from the main results, rather than simply extracted from the result sections, or it was not even possible to derive at all. We have done our best to derive relevant information and avoid misinterpretations from the included studies.
- The review summarises only reported research findings. There are many important aspects with no reported findings. This means that the review cannot be comprehensive regarding what is most important for the real-world front-end phases. Instead, the review will inevitably be biased toward presenting results dominated by the interests of the researchers and data availability. This is similar to most other literature reviews of this type and is mainly a problem if there are highly important aspects with low research interest. While this may be the case for some of our analyses, we believe we nevertheless were able to cover a range of important analyses in our systematic review.
- To claim causation, and not just correlation, between what is done in the front-end phase and the outcome of the agile software development is questionable. To be able to do this, in the lack of controlled experiments, we need to understand the underlying mechanisms and how they are likely to connect the elements for a variety of contexts causally. This has, in most cases, been difficult. The identified connections, even when the same connection is reported in many papers, should consequently often be understood as a potential causal connection, with a need to understand better the underlying mechanisms, the strength of the mechanism, and the necessary and sufficient conditions for the mechanism to take place. This is an inherent

problem for studies of industrial processes where controlled experiments are not possible and there are many potential confounding variables. We address this limitation by discussing possible causal effects and being careful about concluding about causality.

Despite the above limitations and challenges, we found it worthwhile to complete the review. The main reason for this was that we judged several of the identified patterns in results to be sufficiently relevant and potentially important to defend reporting. Our hope is that the findings will support an improved front-end phase for agile software development and inspire more and improved research on this important phase.

7. Conclusions

Changing to an agile way of developing software does not remove the need for requirement specification, estimation of cost and benefits, and work planning before the actual development of the software starts. This need for front-end activities is typically motivated by the wish to know whether an investment in software development is worthwhile or not and to establish an initial plan for allocating resources and managing the work. Perhaps a bit more surprising is our observation that the front-end phase of agile software development tends to be just as comprehensive and detailed as that of non-agile software development. We interpret this observation as suggesting that more than two decades of agile software development has not had much impact on the front-end phase.

This observed lack of influence from agile software development on the front-end phase may be a consequence of software development governance principles that do not sufficiently consider the contextual factors of software development, including the planned level of agility in the software development. A situation with little or no consideration of the actual need for detail and comprehensiveness of the front-end phase activities may be particularly unfortunate for agile software development. This may, according to our review results, be the case as less detail in specifications, estimates, and plans is observed to be connected with more successful agile software development.

Based on the studies reported in our review, it proved challenging to identify and recommend the optimal level of front-end analysis, estimation, and planning for agile software development. The main cause for this is that we do not have a good understanding of the complex network of causal mechanisms connecting elements of the front-end phase and the outcome of agile software development. This challenge is in particular present when interpreting the observation that shorter front-end phases tend to be associated with more success in agile software development. While there are reported findings supporting the presence of mechanisms connecting less detail in the front-end phase with more success, e.g., through enabling more software development flexibility, there are also observed mechanisms in the opposite direction, e.g., that insufficient requirement analysis led to project problems. It may also be the case that a shorter front-end phase is connected with less complex software development work and that the main mechanism is between lower complexity and more outcome success, not between a shorter front-end phase and outcome success. More research, such as large-scale surveys adjusting for the effect of a larger set of potentially confounding variables and in-depth single case studies of the underlying causal mechanisms, are needed to better understand this.

Based on the reviewed studies, it is somewhat easier to defend that there *is* an optimal level of front-end phase comprehensiveness and that this level is dependent on the context of the software development. In particular, the riskiness of the software development seems to be an important contextual factor. An implication of this observation is that low-risk agile software development tends to spend too much time in the front-end phase and that it is advisable to consider this and other contextual factors when deciding on the activities and comprehensiveness of the front-end phase.

Several front-end elements were reported, either statistically or as perceived by the stakeholders, to be connected with a higher level of success in agile software development. Central implications of these findings for agile software development, as we interpret them, are that:

- The work on requirements, estimates of costs and benefits, and work plans in the front-end phase should support flexible execution of the software development. This may imply avoiding highly detailed requirements, estimates and plans, unless more detail is needed to reduce the risk of substantial software development execution problems. This may also imply the use of contracts based on hourly payment, instead of fixed price.
- The steps and comprehensiveness of the front-end phase should reflect the context of the software development. Contextual factors of importance leading to less need for a comprehensive front-end phase include low software development complexity, high domain and technical competence of the software development resources, and high client and user competence.
- The cost estimation in the front-end phase may benefit from the use of a company-tailored checklist for the cost estimation work and the use of relative cost estimation.
- The front-end phase may benefit from experimentation, e.g., with different alternative solutions, to increase understanding of the needs and improve the estimates and planning of the software development.
- Proper front-end phase formulation of the planned benefits from the proposed software development and the development of realistic plans on how to realise these benefits are central elements to enabling good benefits management and successful realization of benefits. The benefits management responsible should, as with the cost estimation, be aware of the potential negative effect of detailed and highly specific business cases, as it may limit the freedom and creativity of agile software development, e.g., identifying new types of benefits and removing previous ones during the software development.

Similarly to our relatively meagre knowledge about the optimal comprehensiveness of the front-end phase of agile software development, our knowledge about how much and in which ways different front-end phase process elements, including the ones presented above, affect the success of agile software development. To gain more robust and useful knowledge, we need more research studies. Such studies should, we believe, aim at including more of the agile software development context and also at understanding the causal mechanisms rather than just simply calculating the correlations between variables or asking software professionals about what they perceive are the connections in their particular context.

CRediT authorship contribution statement

Magne Jørgensen: Writing – review & editing, Writing – original draft, Validation, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this article.

Data availability

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

References

Abrahamsson, P., O. Salo, J. Ronkainen O.G.J. Warsta (2017). Agile software development methods: review and analysis. arXiv preprint arXiv:1709.08439.

Bechtel, J., Kaufmann, C., Kock, O.G.A., 2021. Agile projects in nonagile portfolios: how project portfolio contingencies constrain agile projects' teamwork quality. IEEE Trans. Eng. Manag. 69 (6), 3514–3528.

Bilgaiyan, S., Sagnika, S., Mishra, S., Das, O.G.M., 2017. A systematic review on software cost estimation in agile software development. J. Eng. Sci. Technol. Rev. 10 (4), 52–64.

Blackburn, J.D., Scudder og, G.D., Van Wassenhove, L.N., 1996. Improving speed and productivity of software development: a global survey of software developers. IEEE Trans. Softw. Eng. 22 (12), 875–885.

Boehm, B., 1996. Anchoring the software process. IEEE Softw. 13 (4), 73–82.

Choo, A.S., 2014. Defining problems fast and slow: the u-shaped effect of problem definition time on project duration. Prod. Oper. Manag. 23 (8), 1462–1479.

Cohen, D., Lindvall, M., Costa, O.G.P., 2003. Agile software development. Dacs Soar Rep. 11, 2003.

Cohn, M., 2005. Agile Estimating and Planning. Pearson Education.

Dima, A.M., Maassen, O.G.M.A., 2018. From Waterfall to Agile software: development models in the IT sector, 2006 to 2018. Impacts on company management. J. Int. Stud. (2071-8330) 11 (2).

Dybå, T., Dingsøyr, O.G.T., 2008. Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. 50 (9–10), 833–859.

Edison, H., Wang, X., Conboy, O.G.K., 2021. Comparing methods for large-scale agile software development: a systematic literature review. IEEE Trans. Softw. Eng. 48 (8), 2709–2731.

Edkins, A., Geraldi, J., Morris, P., Smith, O.G.A., 2013. Exploring the front-end of project management. Eng. Project Organ. J. 3 (2), 71–85.

Fernández-Diego, M., Méndez, E.R., González-Ladrón-De-Guevara, F., Abrahão, S., Insfran, O.G.E., 2020. An update on effort estimation in agile software development: a systematic literature review. IEEE Access 8, 166768–166800.

Furulund, K.M., Moløkken-Østvold, O.G.K., 2007. Increasing software effort estimation accuracy using experience data, estimation models and checklists. In: Seventh International Conference on Quality Software (QSIC 2007). IEEE.

Halkjelsvik, T., Jørgensen, O.G.M., 2012. From origami to software development: a review of studies on judgment-based predictions of performance time. Psychol. Bull. 138 (2), 238–271.

Hannay, J.E., 2021. Benefit/Cost-Driven Software Development: With Benefit Points and Size Points. Springer Nature.

Hoda, R., Salleh, N., Grundy, O.G.J., 2018. The rise and evolution of agile software development. IEEE Softw. 35 (5), 58–63.

Jørgensen, M., 2014. Communication of software cost estimates. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering.

Jørgensen, M., 2019. Relationships between project size, agile practices, and successful software development: results and analysis. IEEE Softw. 36 (2), 39–43.

Jørgensen, M., 2023. What Can We Learn from Surveys On the Importance of Software Development Productivity factors? Anais do XXVI Congresso Ibero-Americano Em Engenharia de Software. SBC.

Kupiainen, E., Mäntylä, M.V., Itkonen, O.G.J., 2015. Using metrics in agile and lean software development–a systematic literature review of industrial studies. Inf. Softw. Technol. 62, 143–163.

Lappi, T., Aaltonen, O.G.K., 2017. Project governance in public sector agile software projects. Int. J. Manag. Projects Bus. 10 (2), 263–294.

Larsen, A.S.A., Volden, G.H., Andersen, O.G.B., 2021. Project governance in state-owned enterprises: the case of major public projects' governance arrangements and quality assurance schemes. Adm. Sci. 11 (3), 66.

Laukkanen, E., Itkonen, J., Lassenius, O.G.C., 2017. Problems, causes and solutions when adopting continuous delivery—a systematic literature review. Inf. Softw. Technol. 82, 55–79.

Mahnič, V., Hovelja, O.G.T., 2012. On using planning poker for estimating user stories. J. Syst. Softw. 85 (9), 2086–2095.

Meier, S.R., 2008. Best project management and systems engineering practices in the preacquisition phase for federal intelligence and defense agencies. Project Manag. J. 39 (1), 59–71.

Molokken, K., Jorgensen, O.G.M., 2003. A review of software surveys on software effort estimation. In: 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings. IEEE.

Nguyen, T.S., Mohamed, O.G.S., 2020. Interactive effects of agile response-to-change and project complexity on project performance. In: The 10th International Conference on Engineering, Project, and Production Management. Springer.

Rainer, A., Shepperd, O.G.M., 1999. *Re*-planning for a successful project schedule. In: Proceedings Sixth International Software Metrics Symposium (Cat. No. PR00403). IEEE.

Serrador, P., Pinto, O.G.J.K., 2015. Does Agile work?—A quantitative analysis of agile project success. Int. J. Project Manag. 33 (5), 1040–1051.

Serrador, P., Turner, O.G.R., 2015. What is enough planning? Results from a global quantitative study. IEEE Trans. Eng. Manag. 62 (4), 462–474.

Sudarmaningtyas, P., Mohamed, O.G.R., 2021. A review article on software effort estimation in agile methodology. Pertanika J. Sci. Technol. 29 (2), 837–861.

Usman, M., Mendes, E., Börstler, O.G.J., 2015. Effort estimation in agile software development: a survey on the state of the practice. In: Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering.

Usman, M., Mendes, E., Weidt, F., Britto, O.G.R., 2014. Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th international conference on predictive models in software engineering.

Vyas, M., Bohra, A., Lamba, C., Vyas, O.G.A., 2018. A review on software cost and effort estimation techniques for agile development process. Int. J. Recent Res. Aspects 5 (1), 1–5.

Wang, L., Vidgen, O.G.R., 2007. Order and chaos in software development: a comparison of two software development teams in a major it company. In: 15th European Conference on Information Systems.

Williams, T., Vo, H., Samset, K., Edkins, O.G.A., 2019. The front-end of projects: a systematic literature review and structuring. Prod. Plan. Control 30 (14), 1137–1169.

Zwikael, O., 2020. When doesn't formal planning enhance the performance of government projects? Public Admin. Q. 44 (3), 331–362.

Zwikael, O., Gilchrist, O.G.A., 2021. Planning to fail: when is project planning counterproductive? IEEE Trans. Eng. Manag. 70 (1), 220–231.

Magne Jørgensen is a researcher at the Simula Metropolitan Center for Digital Engineering, a professor at Oslo Metropolitan University, and a consultant and advisor for software companies. His research focuses on software development management, benefits management, agile software development, software cost estimation and software development productivity.