Insights on Agile Contracting: Bridging Theory and Practice

Bert de Brock[1], Konstantinos Tsilionis[2], and Aleksis Mogilnijs[3]

[1] *University of Groningen, PO Box 800, Groningen, 9700 AV, The Netherlands*
[2] *Eindhoven University of Technology, De Zaale, Eindhoven, 5612 AR, The Netherlands*
[3] *KU Leuven, Address, Warmoesberg 26, Brussels, 1000, Belgium*

Abstract

Agile software development, characterized by customer collaboration, evolving requirements, and value-driven features, poses certain challenges in setting up an efficient contracting process at the start of the development project. Despite the increasing number of organizations transitioning to the agile way of commissioning software, there is still lack of comprehensive research in effective contracting practices between clients and developers. In response, this paper presents the findings of a literature review in terms of identifying prevalent contracting practices for agile software projects. Such literature findings are then compared and contrasted to practitioners experiences in the field where concrete suggestions are given. Finally, we sketch how a coherent agile contract could look like.

Keywords

Agile Contracting, Agile Software Development Projects, Literature Review, Practitioners' Perception, Changing Requirements, Customer Collaboration, Understanding Agility, Trust, Risks, Flexible Contracts, Agile Planning.

1. Introduction

Agile software development emphasizes flexibility, customer collaboration, and delivering business value in short cycles [1, 2]. Despite its advantages, challenges such as scalability, team coordination, and the need to establish solid contracts for agile software development projects remain impactful [3, 4, 5, 6, 7, 8]. The last point is considered a major bottleneck since the need to set-up contracts at the start of software development projects seems to be one of the greatest challenges for agile development techniques which promote flexibility at any given moment in the duration of the project [9, 10, 11]. For this reason, this paper presents a compilation of best practices that can be attributed to agile contracting processes through a literature review. It then compares and contrasts such findings to practitioners' perspectives, aiming to explore the alignment (or divergence) between what is considered to be a reasonable agile contracting practice in literature theory and what practitioners actually experience in the field. Finally, based on our findings, we discuss how a coherent agile contract could look like. Therefore, our Research Questions are:

Main Research Question (MRQ)
What are the prevalent issues in the establishment of agile contracts, and
the contract types that are considered effective in managing project scope, timelines, and deliverables while fostering collaboration and flexibility between clients and development teams?

This MRQ can be broken down into several Specific Research Questions (SRQs):

SRQ1: What are the key issues encountered in the set-up of agile contracts?
SRQ2: Which agile contract types are frequently found both in the literature and in practice?
SRQ3: What should be the most important ingredients of an agile contract template?

Agil-ISE24: 3rd Intl. Workshop on Agile Methods for Information Systems Engineering, June 3, 2024, Limassol, Cyprus
EMAIL: e.o.de.brock@rug.nl (A. 1); k.tsilionis@tue.nl (A. 2); aleksis mogilnijs@kuleuven.be (A. 3)
ORCID: 0000-0001-9702-6941 (A. 2)

2. Research Approach

Our methodology combines a literature review with a qualitative research study involving expert opinions on agile contracting practices. The literature review follows the paradigm of Kitchenham & Charters [12] in search for studies related to the most renowned agile contracting practices. Indeed, it stipulates the application of inclusion criteria (e.g., including peer-reviewed studies addressing specifically contracting processes for agile software development), exclusion criteria (e.g., studies unrelated to software development, and written in a language other than English), and the identification of key search terms such as *"software development", "Agile method", "contracting in Agile"*.

Overall, the literature review culminates in 28 papers deemed relevant to study the key issues, contract types, facilitating contract components, and impacting factors that are encountered in contemporary agile contracting processes. Due to the lack of space, these studies are not explicitly mentioned here. However, the study of Mogilnijs[2] et al. [13], on which the present study is based, provides full explications of (i) the search algorithm for the conduct of the literature review, and (ii) the retrieved studies. Next, the results of the literature review are being fed as input to the performance of a qualitative research study entailing the conduct of semi-structured interviews with 8 experts in agile contracting practices, to verify the veracity of the literature sources from a practical perspective. Once again, the reader can find a detailed description of the sampling technique, interview protocol, and profiles of the 8 agile practitioners for the qualitative research study in [13]. It is important to mention at this point that the study of Mogilnijs et al. [13] offers a comprehensive analysis of the essential considerations, influencing factors, and inherent challenges involved in structuring agile projects. On the other hand, the present study aims to concentrate on specific issues that could potentially impede the establishment of agile contracts (see SRQ1), and conceptually exemplify specific types of agile contracts commonly encountered in both literature and practice (see SRQ2). This is performed by presenting the results of the aforementioned literature review and qualitative research. In contrast to [13], lacking concrete (and practical) suggestions for improving the structure of agile contracts, the current paper strives to provide a preliminary outline of what a cohesive agile contract might entail (see SRQ3).

In addressing SRQ1 (*What are the key issues encountered in setting up agile contracts?*), the literature reveals 7 key issues, while the interviews with practitioners identify an additional 8 issues. However, a meta-analysis conducted on these identified issues, utilizing a prominent book on agile project management practices (see [14]), determines that 3 of these issues, which are also rarely mentioned by practitioners, are not directly pertinent to the essence of software agility. Consequently, these issues are excluded from the findings presented in the current paper. The final compilation of the encountered issues in setting up agile contracts is presented in Tables 1 and 2 (see Section 3.1).

In addressing SRQ2 *(Which agile contract types are frequently found both in the literature and in practice?*), the literature reveals 12 contract types, while the interviews with practitioners identify 6 contract types. Applying the same level of meta-analysis as before, we exclude 7 from the 12 contract types mentioned in the papers as they were mentioned infrequently (at most 3 times) in the 28 retrieved sources, or were not directly related to agility, or were limited to only some payment scheme aspect in the contracting process for the software development project (e.g., '*Payment is delayed until …*'). The final compilation of the retrieved contract types frequently found in the literature and in practice for agile software development projects is presented in Tables 3 and 4 (see Section 3.2).

Finally, in order to address SRQ3 (*What should be the most important ingredients of an agile contract template?),* we provide several practical suggestions for improving existing agile contracts. These recommendations (see Sections 3.3, 3.4, and 3.5) draw from our literature findings and real-world examples from the professional journey of a senior agile practitioner (detailed in [14]).

3. Results

This section is structured in terms of answering each of the 3 SRQs stated before. In particular, Section 3.1 will be presenting and discussing the key issues, identified both in the literature and by the agile experts, that challenge the set-up of agile contracts. Section 3.2 will be presenting and discussing

[2] At the time of drafting the present paper, the study of Mogilnijs et al. [13] was in the process of submission in another publication venue.

the contract types that are found both in the literature and by the agile experts. Sections 3.3 and 3.4 will be correspondingly discussing agile contracting and agile planning in more detail. Next, Section 3.5 will treat SRQ3: What should be the most important ingredients of an agile contract template?

3.1. Specific Issues Encountered

This section presents the results of our literature review on key issues in setting up agile contracts. In Table 1, the first column mentions the identification number of the recognized issue, the third column presents an explication of the issue, whereas the fourth column mentions the frequency in which the issue is being mentioned in the retrieved sources. In contrast, Table 2 shows the issues regarding the set-up of agile contracts that are most commonly recognized by agile practitioners. In the tables themselves, corresponding issues are attributed the same letter (e.g., 'A' stands for the customer collaboration / involvement issue in Tables 1 and 2).

Table 1. The frequencies of key issues identified *in the bibliography* in the set-up of contracts for agile projects (Adapted from [13])

ID		Identified Issue	Freq.
1.1		Insufficient trust between the contractor and the client	7
1.2		Focus on rules rather than collaboration	6
1.3		Risk-Sharing	6
1.4	A	Insufficient Customer Involvement	6
1.5	B	Changing Requirements	5
1.6	C	Client demanding fixed price, fixed scope, and fixed deadline	4
1.7	C	No Fixed Price/Budget	4

The primary issue in setting up contracts for agile software development projects often involves insufficient trust between the client and the contractor [15, 16] which hinders collaboration amongst the aforementioned parties [16, 17]. Another issue highlighted in literature is that traditional software project contracts tend to focus excessively on rules rather than fostering collaboration essential for agile development [18, 19]. Customer involvement is also frequently cited, as ensuring active participation and feedback from clients are factors often not formally indicated in contracts [15, 20]. Additionally, there's often a discrepancy in risk sharing between the contractor and the client, with customers seeking binding contracts to minimize risk [16, 21]. Changing requirements is another significant issue that introduces ambiguity in drafting agile contracts [15, 21, 22], making it challenging for contractors to meet client expectations while maintaining stable contracts throughout the project. While clients may appreciate the flexibility to add or modify requirements during the development project, agile contractors often find it difficult to strike a balance between meeting client expectations and avoiding frequent contract modifications. This struggle arises from the need to keep clients satisfied while also ensuring that contracts remain stable throughout the project, despite potential changes in requirements. Some potential solutions for this issue are discussed in Section 3.3. Last, the open parameters of contracts, like scope and price, can pose challenges, as clients may hesitate to accept contracts with undefined parameters, despite the iterative nature of agile methodologies [22, 23].

Table 2. The frequencies of key issues recognized *by the interviewees* in the set-up of contracts for agile projects (Adapted from [13])

ID		Identified Issue	Description	Freq.
2.1		Understanding of agility	Lack of client understanding/experience with agile methods	5
2.2	A	Customer collaboration	Lack of customer communication and involvement	5
2.3	C	Fixed price, scope, or time	Customers demanding fixed parameters	4
2.4	B	Changing requirements	No clear vision & unrealistic demands	3
2.5		Turnover management	Workflow delay due to turnover of stakeholders	2

Note the discrepancies between the list of key issues frequently mentioned in the literature and the list of the key issues frequently mentioned by the practitioners. We will discuss those issues, starting with Table 2 and continuing with the issues from Table 1 that were not mentioned by the practitioners.

Issue 2.1, *Understanding of agility*: Beforehand, the contractor must explain to the client the agile way of working, not all software engineering details but especially the advantages and consequences for the client! For instance, that it creates room to deal with the - unavoidably – evolving and changing requirements (issue B), that there is no need for a completely and precisely specified hard scope upfront (no 'fixed scope'), but also the necessity and importance of serious customer involvement, collaboration, and communication (issue A). The agile way of working requires a completely different mindset than the 'traditional' waterfall attitude.

Issue 2.2 (and 1.4), *Customer collaboration*: The text for this issue is mainly taken from [14], published in 2023 under exclusive license to Springer Nature Switzerland AG 2023. Agility requires intensive user involvement, in particular of the (most) knowledgeable people. So, not some newly appointed juniors or trainees as a kind of 'proxies', but (various) domain experts. But those experts are usually the most occupied people in the user organization as well. Therefore, it can be (very) hard to have them sufficiently available. Nevertheless, it is necessary in order to get a good quality of the system to be developed, and for a timely delivery as well. Our advice: Require their sufficient availability by contract (!), as we always do. See Section 3.3. Then the next question is: How to actually enforce it? Well, as a first answer: Instead of *on-site customer(s)* or proxies you should have **on-site developers**, i.e., developers who are always available on the customer site. Moreover, on-site developers provide the possibility of (almost) *continuous validation* and communication, while the customers can just stay at their own work environment. And, as to be expected, the developers can also occasionally drop by, say at the end of the day. And besides that, there are also the joint lunches and coffee machines…

Issue 2.3 (and 1.6 and 1.7), *Fixed price, scope, time*: Actually, these issues are about *contract types*. In each case, it usually refers to the 'traditional' waterfall mindset of a client who is used to (and comfortable with) fixed price, fixed scope, and fixed deadline. These issues point to the client's fixation to have fixed parameters and to the practitioner's frustration about it. Fixed price, time, and scope worked well in the 'old days' when automating a well-established, stable, and well-understood (manual) process with which there were already many years of experience. However, later on, *new* processes in existing organizations had to be automated as well. But such new processes were often not yet well-established and not yet so well-understood. And nowadays you often have to develop information systems for businesses that still have to be developed themselves. So, you then have *concurrent development* of a business (model) and its enabling information systems [14]. We refer to Section 3.3 and our answer to Issue 2.1 how to deal with it.

Issue 2.4 (and 1.5), *Changing requirements*: The mentioned clear vision should come from the contractor and be explained to the client (see Issue 2.1). This might also prevent unrealistic demands from the client. Since requirements might start vague, incomplete, inconsistent, or even wrong, a project might not start with clearly, correctly, and completely formulated problems. Moreover, initial requirements might change during the project, e.g., due to new insights or due to external or internal changes [14]. Hence, the development project should account for that.

Issue 2.5, *Workflow delays*: Workflow (and other) delays can happen at 'any time' and for any reason. Therefore, as contractors, we usually let our developers work at two projects at a time (and two developers at a project), in order to avoid idle time of our employees (and idle time for the project). Since the clients' employees can stay at their own work environment (see Issue 2.2) and usually have other tasks as well, they might not have idle time problems because of delays in the project. Also, timing statements in contracts might contain a clause like 'except / not counting (workflow) delay'.

Issue 1.1, *Insufficient trust*: This probably comes from previous experiences of a client who is used to a 'traditional' waterfall approach, where there is a (hopefully) working system only after much investment in time and money, so the risks are (very) high; see Section 3.2 on project failures as well. Insufficient trust between the contractor and the client may lurk then. We note that this was not an issue mentioned by the practitioners. Instead, they focus on the customer's understanding of an agile way of working as a key issue (Issue 2.1). The lack of trust can be further mitigated by the points in Issue 1.3.

Issue 1.2, *Focus on rules rather than collaboration*: This might be a consequence of - and reaction to - Issue 1.1. So, again, it is important that the contractor explains (and emphasizes) the agile way of working to the client beforehand (see Issue 2.1).

Issue 1.3, *Risk-Sharing*: Instead of *Risk-Sharing*, we emphasize and steer on *Risk Reduction*, or even *Risk Avoidance*. As cited from [14], a system could be developed and delivered 'piece by piece'. How large or small should a 'piece' be? A so-called *module* is a good candidate: By a **module** we mean a '*functional unit of independent utility*' [14]. It could cover the tasks – or one of the tasks – of a (sub)department, for instance. A module might be comparable to a milestone. A module might contain a handful to a dozen of user wishes. Delivery of a module means delivery of a useful, measurable sub-product and an early return on (time) investment (and also a good moment for the contractor to send a bill). Since each time a '*functional unit of independent utility*' is delivered, the money was well spent. Delivery of a module contributes to the relation with the client as well.

3.2. Contract Types Identified

This section demonstrates the contract types for agile software development projects that are frequently identified in our compiled literature review. These results are encapsulated in Table 3. In contrast, Table 4 shows the contract types most commonly recognized by agile practitioners. In both tables, the contract types emphasize the aspects *price*, *time*, and *scope*, and corresponding contract types have been given the same letter. We note that these three aspects correspond to the three basic project requirements from project management theory, which are:

- The project must deliver the required functionality (*scope*)
- The project must be within time (*time*)
- The project must be within budget (*price*)

We mention these elements since, even after decades of numerous bad experiences, many software development projects are still failing on at least one or even all three basic requirements for a project:

- Too little: The project delivers inadequate functionality
- Too late: The project is not within time
- Too costly: The project is not within budget

Or even worse, projects can end prematurely, usually after a lot of time and money has been spent, and without delivering any working functionality [14].

Table 3. The frequencies of contract types identified in the *bibliography* for agile projects (Adapted from [13])

ID		Contract Type	Description	Freq.	Price	Time	Scope
3.1	A	Time and Materials (T&M)	Hourly rate with flexible time and scope.	15	fixed per *hour*	flex	flex
3.2	B	Fixed Price	Fixed price, scope, and time.	15	fixed	fixed	fixed
3.3	C	Payment Per Sprint	Fixed price per sprint.	6	fixed per *sprint*		
3.4	D	Agile Fixed Price	Fixed price and time, open scope.	5	fixed	fixed	open
3.5		Hybrid	T&M with a fixed price per milestone.	5	fixed per *milestone*	flex	flex

Table 4. The frequencies of contract types recognized *by the interviewees* for agile projects (Adapted from [13])

ID		Contract Type	Description	Freq.	Price	Time	Scope
4.1	A	Time and Materials (T&M)	Hourly rate with flexible time and scope	8	fixed per *hour*	flex	flex
4.2	B	Fixed price	Fixed price, scope, and time	5	fixed	fixed	fixed
4.3	C	Pay per sprint	Fixed price per sprint	1	fixed per *sprint*		
4.4	D	Agile fixed price	Fixed price and time, open scope	1	fixed	fixed	open
4.5		Fixed price, open deadline	Fixed price and scope, open deadline	1	fixed	open	fixed

Overall, we find a consensus between literature and practitioners regarding the prevalence of fixed-price and Time and Materials (T&M) contracts. Fixed-price contracts are criticized in the literature for limiting scope changes, emphasizing price over quality, and leading to less frequent feature delivery, issues that might potentially cause conflicts [24, 23, 21, 25]. On the contrary, practitioners highlight the flexibility of T&M contracts in accommodating agile methodologies, with some customizing them to ensure continuous engagement and customer involvement [26]. Despite the literature's praise for pay-per-sprint contracts for faster payment cycles and alignment with Scrum projects [27, 28, 15], practitioners remain confident in using T&M contracts. Similarly, Agile fixed-price contracts, which offer flexible scope and high-level outcome specifications, don't diminish practitioners' preference for T&M contracts [29, 30]. Last, we note we could not find any concrete mention in the literature and the interview data on the *Time* and *Scope* parameters of the contract type C (*Pay per sprint*).

3.3. Views On Agile Contracting

We now discuss *Agile Contracting* and work out a concrete, workable solution as we used in practice and as cited from [14]. How could an *agile contract*, i.e., a contract for an agile development project, be formulated? In an agile development project, the requirements are not completely spelled out beforehand, but can be supplemented later on, and can even be changed during development. The customer must have the freedom to change their mind. Consequently, you cannot concretely specify the deliverables beforehand. So, for agile projects, a fixed scope (underlined in the tables) is unnatural. And fixed (price) contracts (B: fixed price, time, *and* scope) even sound 'waterfall-like'.

Although the customer must have the freedom to change their mind, the customer's requirements should not jump around all the time. They have to 'converge'. The need for their 'convergence' led us to the contract clause (as accepted by the client) that '*It is the intention that the set of requirements converges to an end point and that the system under development converges to the final set of requirements*' [14]. This is related to the issues 1.5 and 2.4 (Changing Requirements).

Furthermore, it could be stipulated *on contract level* to work by means of a series of intermediate versions of the system under development (say, one version per quarter), as well as a series of intermediate versions of a set of requirements (see next paragraph too). This leaves room for a *daily practice* of working with short sprints and even 'nightly builds'. *On contract level*, we also use the clause '*This way of working assumes that the right persons are sufficiently available for deliberation*' [14], which is related to the issues 1.4 and 2.2 (Insufficient Customer Involvement).

The agile planning description here is mainly cited from [14]: We stipulated that in order to achieve the intended convergence, the *deliverables* of each Milestone N will be:

1. the agreed result
2. a detailed set of requirements for Milestone N + 1
3. a global set of requirements for Milestone N + 2

As a start, the contract also specified the agreed result for the first milestone and the set of requirements for the second milestone. Furthermore, the contract stipulated that the detailed set of requirements for the next milestone and the global set of requirements for the milestone thereafter will always be drawn up in mutual deliberation with the client. So, it is a kind of '*rolling specification*'. Upon a client's acceptance of a deliverable, a bill could be sent (as formulated in the contract) and the project continues (unless there is a severe reason to stop then).

3.4. Views On Agile Planning

Planning in agile development is not fixed beforehand – neither predictive nor prescriptive – but *adaptive*. It could consist of three levels of planning [14], for instance:

- <u>Macro</u> Global plan for the entire project (say, a few years):
  - Rough goals, milestones, and time path at a high macro/project level
- <u>Meso</u> High-level plan for the next 'module' (say, for 3 months). For instance:
  - Goals, milestones, and time path for the next quarter ('quarter n + 1')

  - Global goals for the quarter thereafter (‘quarter n + 2’)
- <u>Micro</u> Concrete plan for the next iteration (say, for 1 or 2 weeks)
  - Concrete goals and time path for the next iteration

So, planning always stays one stage ahead of the work in progress. Some sort of ‘*rolling planning*’ makes it easy to adapt the planning to new circumstances. For example, [14] describes a project for a rapporteur of the *European Medicines Agency*[3] (EMA) who was in an advanced ideation stage on how to improve the quality of their EMA-reports. All of a sudden, the EMA (comparable to the U.S. Food and Drug Administration) announced a new ‘*Summary of Efficacy Table*’ template for the so-called *Day 80 (assessment) report*. Until then we never even heard of a ‘Day 80 Report’. Nonetheless, after some explanation by the stakeholders, it was very clear that this was a golden opportunity to draw attention to – and to promote – the system under development (the project mentioned in [14]). This EMA-announcement came in the middle of the execution of a quarterly (meso-level) plan. It was decided to finish that meso-level plan as foreseen, because that clearly had its merits too, but to change the plans for the next quarter(s) completely, such that the system would as soon as possible be able to generate the contents of a ‘*Summary of Efficacy Table*’ according to the new Day 80 Report template.

3.5. The most important ingredients of an agile contract template

Based on the findings previously mentioned in this paper and the first author’s decades of experience with contracts in IS development for external customers, this section puts all the desirable contract ingredients together. We sketch a coherent practical solution which mitigates or even prevents the problems mentioned in this paper. It is in line with what the other practitioners say.

First of all, since the agile way of working requires a completely different mindset than the ‘traditional’ waterfall attitude, the contractor must explain the agile way of working to the client beforehand, especially the advantages and consequences for the client. For instance, (a) that it creates the possibility to deal with evolving and changing requirements, (b) that there is no need for a completely and precisely specified scope upfront, (c) the necessity and importance of customer involvement, collaboration, and communication. The contractor should require the sufficient availability of the knowledgeable people by contract (‘*This way of working assumes that the right persons are sufficiently available for deliberation*’). On the other hand, the contractor should provide **on-site developers**.

During the development project, requirements will become more specific, evolve, and change. However, the requirements should not jump around all the time. A contract clause could be something like ‘*It is the intention that the set of requirements converges to an end point and that the system under development converges to the final set of requirements*’.

The contract could also stipulate to work by means of a series of intermediate versions of a set of requirements and of intermediate versions of the system under development (*Milestones*). This leaves room for a *daily practice* to work with short sprints and even ‘nightly builds’.

The *deliverables* of a Milestone could be: (1) the agreed result, (2) a detailed set of requirements for the next Milestone, and (3) a global set of requirements for the Milestone thereafter. As a start, the result for the first milestone and the set of requirements for the second milestone must be described. So, it is a ‘*rolling specification*’. Upon a client’s acceptance of a deliverable, the project continues (unless there is a severe reason to stop). So, it is a (silent) *continuation clause*, not an *early termination clause* [15, 22]. The requirements should be drawn up in mutual deliberation with the client.

Planning in agile development is neither predictive nor prescriptive, but *adaptive*. It could consist of a few levels of planning, for instance:

- <u>Macro</u> Global plan for the entire project (say, a few years): Rough goals, milestones, and time path.
- <u>Meso</u> High-level plan for the next ‘module’/milestone (say, for 3 months). For instance: Detailed set of requirements and time path for the next milestone, global goals for the milestone thereafter
- <u>Micro</u> Concrete goals for the next iteration/sprint (say, for 1 or 2 weeks)

So, planning always stays one stage ahead of the work in progress. Such a ‘*rolling planning*’ makes it easy to adapt the planning to new circumstances.

[3] More information about the European Medicine Agency is given here: https://www.ema.europa.eu/en

Our approach steers on *Risk Reduction*, or even *Risk Avoidance*, instead of *Risk-Sharing* (Issue 1.3): A system could be developed and delivered ‘module by module’, where a **module** is a ‘*functional unit of independent utility*’. A module might be comparable to a milestone. Delivery of a module means delivery of a useful, measurable sub-product and an early return on investment (of time and money). Delivery of a module also contributes to the trust of the client in the contractor (Issue 1.1). So, the focus should and can be on collaboration rather than on rules (Issue 1.2). Finally, delivery of a module is also a natural occasion for payment.

Avoiding idle time: All kinds of delays can happen at any time and for any reason. Since the clients’ employees can stay at their own work environment (see Issue 2.2), they might not have idle time because of project delays. Contractors could let their developers work at 2 projects at a time (and 2 developers at a project), in order to avoid idle time of their employees (and idle time for the project). Timing statements in contracts might contain a clause like ‘except / not counting (workflow) delay’.

4. Conclusions

The present paper aimed to compile the most cited factors that influence the drafting processes for agile software development projects. Overall, the results show that while academic literature advocates the case for trust-building and use of collaborative contracts, practitioners often resort to traditional contracting methods (e.g., fixed-price contracts) due to client skepticism or inexperience to agile ways of working. Overall, there seems to be a consensus among experts favoring fixed-price and T&M contracts over other types. However, the former types seem to be often customized in practice to accommodate for the agile project deliverables. The impression might be that the reasoning is often how to turn a ‘classical waterfall contract’ into an ‘agile-like’ contract. But, in our opinion, one should not ‘update’ a ‘waterfall contract’ but design an ‘agile-like’ contract from scratch. The agility clauses must form a coherent and complete set of clauses, e.g., as we sketched in Section 3.5.

References

[1] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439.

[2] Tsilionis, K., Wautelet, Y., & Tupili, D. (2021). Digital Transformation and Operational Agility: Love Story or Conceptual Mismatch. In PoEM Workshops (pp. 1-14).

[3] Hajjdiab, H., & Taleb, A. S. (2011). Adopting Agile software development: Issues and challenges. International Journal of Managing Value and Supply Chains, 2(3), 1–10.

[4] Ktata, O., & Lévesque, G. (Eds.). (2009). Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. ACM International Conference Proceeding Series, 59–66.

[5] Shafiee, S., Wautelet, Y., Hvam, L., Sandrin, E., & Forza, C. (2020). Scrum versus Rational Unified Process in facing the main challenges of product configuration systems development. Journal of Systems and Software, 170(1), 1–23.

[6] Snoeck, M., & Wautelet, Y. (2022). Agile MERODE: a model-driven software engineering method for user-centric and value-based development. Software and Systems Modeling, 21(4), 1469–1494.

[7] Tsilionis, K., & Wautelet, Y. (2022). A model-driven framework to support strategic agility: Value-added perspective. Information and Software Technology, 141, 106734.

[8] Tsilionis, K., & Wautelet, Y. (2021). From service-orientation to agile development by conceptually linking business IT services and user stories: A meta-model and a process fragment. In 2021 IEEE 23rd Conference on Business Informatics (CBI) (Vol. 2, pp. 153-162). IEEE.

[9] Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. Computers in human behavior, 51, 915-929.

[10] Tsilionis, K., Maene, J., Heng, S., Wautelet, Y., & Poelmans, S. (2021). Conceptual modeling versus user story mapping: Which is the best approach to agile requirements engineering?. In International Conference on Research Challenges in Information Science (pp. 356-373). Cham: Springer.

[11] Tsilionis, K., Maene, J., Heng, S., Wautelet, Y., & Poelmans, S. (2021). Evaluating the software problem representation on the basis of rationale trees and user story maps: premises of an experiment. In Software Business: 11th International Conference, ICSOB 2020, Karlskrona, Sweden, November 16–18, 2020, Proceedings 11 (pp. 219-227). Springer International Publishing.

[12] Kitchenham, B., Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering, technical report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, Keele, 9 July.

[13] Mogilnijs, A., Tsilionis, K., Wautelet, Y., & Scharff, C. (2024). A Mixed-Methods Approach to Evaluate Contracting Choices in Agile Software Development Projects. In The International Research & Innovation Forum 2024, The twin transition: leveraging breakthrough technologies & sustainability for innovation, quality education & policy making (Submitted).

[14] De Brock, E.O. (2023). Developing Information Systems Accurately - A Wholistic Approach. Springer Cham.

[15] Zijdemans, S. H., & Stettina, C. J. (2014). Contracting in Agile software projects: State of art and how to understand it. In G. Cantone & M. Marchesi (Eds.), Agile Processes in Software Engineering and Extreme Programming (Vol. 179, pp. 78–93). Springer International Publishing.

[16] Dunbar, I., & Taylor, K. (2007). Agile approaches to knowledge transfer. Engaging HEIs in Business and the Community, 1–21.

[17] Thorup, L., & Jensen, B. (2009). Collaborative Agile contracts. 2009 Agile Conference, 195–200.

[18] Antinyan, V. (2023). Seven lessons from seven automotive software supplier collaborations. IEEE Software, 40(1), 77–85.

[19] Aoufi, A., Schoeman, M., & Turner, N. (2022). How to outsource Agile projects effectively. Research-Technology Management, 65(1), 59–66.

[20] Jørgensen, M. (2016). A survey on the characteristics of projects with success in delivering client benefits. Information and Software Technology, 78(1), 83–94.

[21] Hussein, B. A., & Siddique, L. (2016). Grounded theory study of the contracting process in Agile projects in Norway's software industry. Journal of Modern Project Management, 4(1), 53–62.

[22] Pries-Heje, L., & Pries-Heje, J. (2014). Agile contracts: Designing an agile team selection guideline. Selected Papers of the Information Systems Research Seminar in Scandinavia, 5(1), 34–49.

[23] Hoda, R., Noble, J., & Marshall, S. (2009). Negotiating contracts for Agile projects: A practical perspective. In P. Abrahamsson, M. Marchesi, & F. Maurer (Eds.), XP 2009: Agile Processes in Software Engineering and Extreme Programming (Vol. 31, pp. 186–191). Springer.

[24] Beulen, E. (2019). Implementing and contracting Agile and DevOps: A survey in the Netherlands. Digital Services and Platforms. Considerations for Sourcing, 344(1), 124–146.

[25] Mohagheghi, P., & Jørgensen, M. (2017). What contributes to the success of IT projects? Success factors, challenges and lessons learned from an empirical study of software projects in the Norwegian public sector. 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 371–373.

[26] Siddique, L., & Hussein, B. A. (2016). Grounded theory study of conflicts in Norwegian Agile software projects: The project managers' perspective. Journal of Engineering, Project, and Production Management, 6(2), 120–135.

[27] Davies, C. (2019). Agile contracting for IT services - myth or reality? Communications Law, 24(2), 56–61.

[28] Franklin, T. (2008). Adventures in Agile contracting: Evolving from time and materials to fixed price, fixed scope contracts. Agile 2008 Conference, 269–273.

[29] Banerjee, U., Narasimhan, E., & Kanakalata, N. (2011). Experience of executing fixed price off-shored Agile project. ISEC '11: Indian Software Engineering Conference, 69–75.

[30] Gerster, D., & Dremel, C. (2020). How agility can be increased in IT sourcing and contracting: Learnings from an autonomous driving case. In I. Oshri, J. Kotlarsky, & L. P. Willcocks (Eds.), Digital Technologies for Global Sourcing of Services, 410, pp. 17–37. Springer