Scrum Manager®
Uncovering better ways

DISCOVERING *AGILE CONTRACTS*

AN INTRODUCTION TO AGILE CONTRACTS

New legal models for new paradigms

Version 2 (2026)

Discovering Agile Contracts

An Introduction to Agile Contracts

Version 2 (2026)

Scrum Manager ®

Version date: August 2026.
Version author: the Scrum Manager® team.

Table of contents

INTRODUCTION

Purpose and contents of the study.

Agile project management is defined by its flexibility—its ability to adapt to the needs of constantly changing markets. That very flexibility complicates the drafting of contracts, in which client and supplier each try to protect themselves, seeking maximum benefit at minimum risk.

This research sets out to clear up questions about agile contracts and how they are written: what they are, what they aim to achieve, and the principles behind them. It starts from traditional contracts in order to examine why they can fall short on certain projects.

Chapter 1, "Project Management," lays out some basic ideas about the difference between traditional and agile management—the groundwork for the chapters that follow.

Chapter 2 then defines what agile contracts are, with an emphasis on understanding the needs they address. We recommend reading it before moving on to Chapter 3, which looks at how to create agile contracts.

Chapter 4 brings together all the models and clauses considered agile today. For easy reference, they are arranged alphabetically, each with a short description and the advantages and drawbacks it may carry depending on the circumstances.

Finally, Chapter 5 goes deeper into strategies and tools for negotiating and applying agile contracts, drawing on the sources consulted.

CHAPTER I

PROJECT MANAGEMENT

An overview of the two most popular approaches to project management today. The traditional model now shares the stage with so-called "agile management".

Traditional management

Until recently, the norm in business was the waterfall model: a predictive, process-oriented way of managing projects built on exhaustive planning. Its legal counterpart is the traditional contract—fixed-price or time-and-materials agreements, for example.

Contracts like these come with a set of limitations that mirror the waterfall model. As Jeff Sutherland puts it:

- **Outcomes carry the wrong incentives.** Effort is rewarded over the value delivered to the project, and because change is constrained, continuous improvement is discouraged and needless complexity creeps in. This erodes trust between the parties and breeds unreasonable expectations.
- **Product value is eroded.** The client doesn't get what they actually want, because the process is specified instead of a clear vision. Rigidity limits room to innovate, and when changes do arise, they spark conflict between the parties and pull attention away from scope. Time and effort are wasted trying to comply with processes locked in at the outset.
- **Change is discouraged.** Changes in predictive projects are more costly, so the supplier tends not to bother looking for alternatives. That hampers learning and the ability to respond to market conditions in time.

Agile management

The origins of agile project management can be traced to practices adopted in the 1980s by companies such as Honda, Canon, Fuji, NEC, HP, and Epson, described by Takeuchi and Nonaka (1986) in their influential article *The New New Product Development Game*. Although it was the software industry that first took it up and first popularized various ways of putting agile principles into practice. Over the following decades, a range of agile frameworks emerged: XP (eXtreme Programming), Scrum, DSDM, Crystal, FDD (Feature-Driven Development), ASD (Adaptive Software Development), and Lean Software Development. Over time the landscape has settled: today Scrum and Kanban are the most widespread frameworks, and scaling frameworks such as SAFe and LeSS have emerged to coordinate multiple teams.

No discussion of agility would be complete without the *Agile Manifesto*. In March 2001, seventeen critics of process-based production models were brought together by Kent Beck for a meeting at which the term "agile methods" was coined. These were the alternative they proposed to traditional methodologies and to the shortcomings they had observed in them. They distilled their ideas into four core values:

- **"Individuals and interactions over processes and tools."** This is the manifesto's most important value: processes should serve only as an operating guide. Tools improve efficiency, but agile projects call for talent and motivated people to provide it.
- **"Working software over comprehensive documentation."** Documentation isn't dismissed as useless—only the unnecessary kind. Legal documents are often mandatory, but they should matter far less than the final product. Communicating through documents creates less value than direct conversation between people and hands-on interaction with product prototypes.
- **"Customer collaboration over contract negotiation."** Agile practices suit products that start out loosely defined, precisely so the end result doesn't lose value. Remember that the goal of an agile project is to deliver the greatest possible value in the product. Collaborating with the customer and earning their commitment is therefore far more effective: it makes communication easier and lets them suggest changes from the very start.
- **"Responding to change over following a plan."** For products with unstable requirements—ones that change and evolve quickly and continuously—responsiveness is far more valuable. Knowing how to anticipate change, or adapt to it, is essential.

From these values come the twelve agile principles:

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Manifesto

Results of agile management

The success of agile projects has led more and more companies to adopt them, to the point where agility has become the dominant way of managing software development. That the approach matters is beyond dispute: in BearingPoint's *Agile Pulse 2022* study, 96% of participants rated its future relevance as high. Wanting it, however, is not the same as achieving it: that same study found that, on average, organizations fail to meet eight of every nine goals they had set when adopting agility (BearingPoint, 2022). The conversation, in other words, has shifted from whether to be agile to how to get real value out of it.

That same tension surfaces in the *18th State of Agile Report* (Digital.ai, 2025): AI adoption in agile environments jumped from 68% to 84% in a single year, yet only 49% of organizations have governance mechanisms in place for it, and 76% report growing pressure to demonstrate the return on their agile investment. These figures are best read as a signal of direction—the 2025 sample was small, around 349 responses—rather than as a precise measure.

The evidence on how the two approaches compare in performance is slipperier than it is usually made out to be. The CHAOS report from The Standish Group (2015) is often cited as showing that agile projects have markedly higher success rates than waterfall ones (Fig. 1). But those figures should be taken with caution: the methodology has been questioned for the difficulty of reproducing its results and for a narrow definition of "success"—meeting schedule, cost, and scope—that penalizes exactly what agility treats as negotiable (Eveleens & Verhoef, 2010).

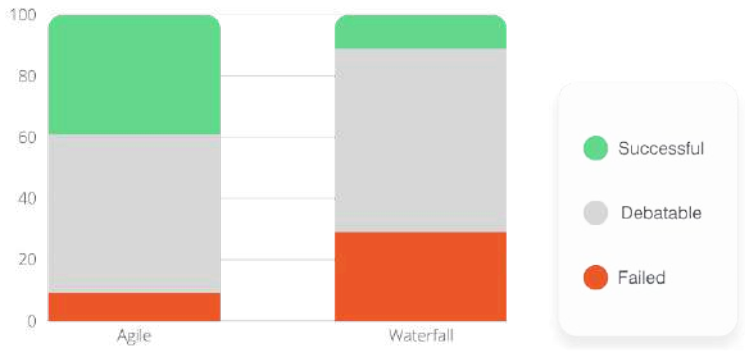

Fig. 1. *Comparison based on data from The Standish Group's CHAOS report (2015), comparing agile and waterfall projects*[1].

This new way of managing work creates a need for new kinds of contracts. Traditional contracts reflect the management models they evolved alongside: predictive, process-oriented, and restrictive toward change. They fit projects that emphasize

[1] Since 1994, The Standish Group has published the CHAOS report, drawing on studies of more than 50,000 projects.

planning and documentation well. They will hardly meet the needs of an adaptive project—one oriented toward people and change.

So if this model is here to stay, suppliers and clients need legal agreements that are consistent with its philosophy.

CHAPTER II

AGILE CONTRACTS

Contracts adapted to follow the principles of the *Agile Manifesto*.

Traditional contracts pose serious difficulties or limitations for companies that use agile methodologies—difficulties that can be summed up as a lack of flexibility. For some authors, such as Opelt and Gloger (2013), agile contracts are a natural evolution of traditional ones, a search for solutions that offer guarantees while allowing for the changes that may arise over the course of a project. And all of this in a way that still fits within the traditional legal framework. What emerges is a synthesis that takes into account, on one hand, the principles of the Agile Manifesto and, on the other, the iron triangle of classic project management. This triangle is a typical representation of the interplay between a project's scope, time, and cost[2].

According to Pries-Heje and Pries-Heje (2014):

- **Scope** refers to the project's objectives described as functionality, usually written as specific requirements.
- **Time** is the project schedule.
- **Cost** is the amount of resources invested.

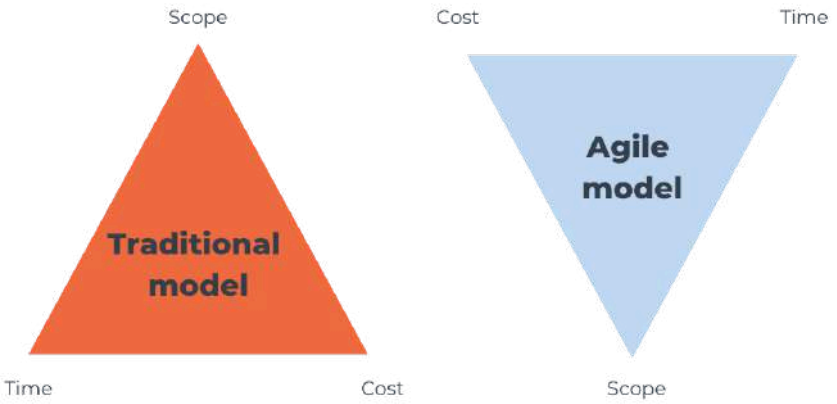

Fig. 2. *On the left, the iron triangle. On the right, the agile alternative.*

The iron triangle is static: the available time and resources bound the scope. In an agile contract, by contrast, at least one of the elements must be variable. Otherwise, it would no longer be agile. Time and cost can vary depending on the scope, which is usually unpredictable in this kind of project.

These are, therefore, **hybrid contracts:** they share features with a traditional contract while also trying to break away from it by betting on **adaptability.** That said, there is no definitive agile contract, just as there is no definitive traditional contract. There is no copy-and-paste template, because each one depends on as many factors as affect each project. Whatever the type of contract, it must suit the circumstances—with the difference that agile contracts do more to anticipate and account for future changes, for example through clauses.

[2] The triangle (or triple constraint) is a classic project-management concept that predates agility; Pries-Heje and Pries-Heje (2014) are cited here for their definitions, not for the origin of the concept.

In short, an agile contract tries to meet the objectives of a legal contract, but looks for other ways to do so. And although there is no magic formula for writing one, it is possible to identify principles common to their structure.

Applied principles

> For a contract to succeed, it must be the result of relationships based on trust, collaboration, and transparency.
>
> Gerster and Dremel (2019, p. 4)

For an agile contract to succeed, it should allow the team to fulfill the twelve principles of the *Agile Manifesto*. As noted, these contracts are hybrid in nature and share elements with traditional ones. One way to understand the principles of an agile contract is as the key points of any business agreement, to which agile principles are then applied.

The following sections develop those points, showing how they are handled in agile contracts and how each adaptation relates to the principles of the *Manifesto*.

Value-based incentives

In traditional contracts, paying for work done can create a conflict of interest between client and supplier. For example: if payment is by time spent, the supplier has no incentive to finish before the delivery date, while the client will want precisely an early delivery to save costs. This creates a tension that hurts the efficiency of the project.

Faced with situations like these, authors such as Jeff Sutherland recommend adjusting the incentives to align the interests of client and supplier (Sutherland, n.d.).

Figure 3 shows a table with the incentives used in the traditional model, the results they can produce, and possible alternatives in agile contracts.

Traditional approach	Undesirable outcomes	Better incentive
Pay by the hour	Rewards spending more time. Lowers the productivity of improvements.	Pay per story point delivered.
Pay by lines of code	Doesn't reward quality. Produces less clean, less concise code.	Pay by business value delivered.
Partially completed functionality	Too many incentives for work in progress.	Count only releasable functionality within the business value delivered.
Fixed delivery date and scope	Encourages teams to do less in order to deliver on time.	Choose one of the two: fixed delivery date or fixed scope.
Complex approval process	Slow deliveries. Experimentation is limited.	Have a single approval point and a process just sufficient to ensure quality.

Fig. 3. *Jeff Sutherland's table of incentives.*

Offering incentives or payment by story points or business value, rather than by hours or lines of code, are clear examples of how these changes align better with the principles of the *Agile Manifesto*—above all with these:

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
>
> Working software is the primary measure of progress.

Through incentives like these, agile contracts reflect the agile philosophy of aiming to deliver value to the client.

Flexibility

Flexibility is fundamental to agile projects—so much so that agile methodologies always include mechanisms for more adaptable development. Building these aspects into the contract, which may vary depending on the methodology used, helps make it flexible and connects with several principles of the *Manifesto*.

- **Establish regular feedback cycles.** For example, specifying that the team will work in sprints, with the meetings this usually entails, and how often backlog refinement sessions will take place. If the supplier meets with the client at short, regular intervals to review new functionality, the client can give feedback and suggest changes. In a traditional model, changes tend to be requested at the end, when the delivery date arrives, driving up cost, time, and effort.
- **Identify roles and processes** for receiving client feedback and refining the backlog. The initial ground rules of the agreement are defined here, recording how changes will be handled and who will manage them (Sutherland, n.d.).

Although these strategies connect with several principles of the *Agile Manifesto*, perhaps the most evident are these:

> Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
>
> Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

These principles promote an iterative, incremental approach that lets teams deliver value early and continuously and cope with change without wasting resources.

Direct collaboration

> Business people and developers must work together daily throughout the project.

In agile projects the client must be fully involved in the development, and this should be reflected in the contract as well.

- **Full client commitment:** the client must be involved, ideally by naming a representative who will collaborate directly with the team during development (the Product Owner role in Scrum). To achieve this, the objectives, each party's involvement, and the risk—which, though high, must be shared—need to be made clear. Most important is to avoid making demands of the client: offer alternatives for reaching the objectives. Pursue whatever delivers the most value and, where there are obstacles, try to reach a resolution. In other words, client and supplier need to help each other.
- **Managing responsibilities:** you have to know how to handle each party's way of working and pace. Perhaps the client is too slow and the supplier needs to offer solutions so they can keep up with the pace of work, since sometimes the client may have more than one project. Ideally, alternatives are found jointly so that this doesn't affect the pace of work (Tortorella, 2012).

Frequent communication

> The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Poor communication or a lack of transparency hampers the project's progress and damages the relationship with the client. For collaboration to flow, both parties need to stay informed at all times, and to that end it is advisable to establish communication routines as early as possible.

In the traditional model, the client communication we are used to is not continuous and not always planned. It tends to produce late feedback and a loss of focus that affects scope and cost.

With an agile methodology, the aim is:

- **Visibility of progress:** the client has to see what is being worked on in order to suggest changes during the process. Knowing what is being done gives a sense of control and a say. To this end, weekly meetings with the client are often written into the contract, or progress is shared daily through an application.
- **Joint user testing:** related to the previous point. In the traditional model, it is common for the client not to test their own product. But, again, if the client doesn't

test it, they can't know whether they want changes or improvements. They will realize later, which means higher cost, effort, and time. The usual practice is to set up sessions dedicated exclusively to testing the product together with the client, and to include them in the contract.

- **Managing expectations:** it is very important to be clear about each party's objectives and how far things can go. For example, the supplier should promise only what it can deliver, and the client should know this as soon as possible. It is a good idea to clarify the project's objectives as early as possible and to collaborate on building a detailed backlog, aware that it will change.

Trust

Agile contracts have a disadvantage compared with the traditional model: it is common for a certain initial distrust to exist when starting to work with a new supplier; and if the client has also never worked with agile methodologies before, that distrust grows.

One solution, similar to what we would do in the traditional model, is to provide references. The more information the client has, the more secure they will feel. Information about other projects carried out in an agile way can be offered. In addition, the marketing and sales departments should make an effort to understand clients' needs and doubts, in order to give them suitable information about this kind of contract.

Even so, this is sometimes not enough. Many companies resort to including certain clauses that provide extra reassurance to reluctant clients. For example, a clause about a possible agreement for a low-risk pilot project or a sprint 0 (Tortorella, 2012).

To improve trust between the parties, other clauses can also be included specifying the consequences of certain behaviors, such as a lack of candor about the state of progress during iterations. This helps prevent damage to the client–supplier relationship (Sutherland, n.d.). It benefits both parties, since a lack of trust or a poor relationship can stress the team, sap motivation, and lower efficiency (Tortorella, 2012).

In truth, every aspect of the contract—from the focus on delivering value to the client to frequent, direct communication—is related in some way to strengthening trust between the parties. It is an intangible without which a project can hardly succeed, since agility rests on self-organized, capable, and motivated teams. As the *Manifesto* says:

> Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Advantages and disadvantages

Advantages and disadvantages of agile contracts in general (Sutherland, n.d.):

Advantages

They allow a legal agreement that is aware of change. They assume that changes are part of development and are able to handle them with confidence, avoiding unpleasant surprises.

They increase efficiency without harming the supplier. Frequent, direct communication, agreed through a suitable contract, makes it possible to solve problems faster and generate real value sooner.

Changes carry lower costs, as a direct consequence of the two previous advantages. They can be made on the fly, before errors grow large enough to hurt the project's budget. This lowers costs and produces a genuinely valuable product for the client.

They boost the supplier's creativity: these contracts provide the flexibility needed to experiment and propose innovative solutions, which can arise during development and not only during the planning phase.

They require less time spent on formal project documentation. The formal requirements documents typical of predictive management are far heavier than the backlogs of agile management.

They tend to produce high-quality products. With no closed scope document but rather a living product backlog, the supplier can suggest changes without fear of derailing the entire scope, and generate more value in less time.

Disadvantages

Initial distrust. If the client doesn't know what agility is, the limited sense of control in the face of possible changes can breed distrust at the outset.

Flight to more like-minded suppliers. That distrust can lead the client to look for other suppliers more aligned with their way of working. An agile supplier may then be tempted to compete with them using the wrong strategies.

Compatibility problems. If development is shared with other, non-agile suppliers, problems can arise from the different ways of working.

Uneven distribution of risk. If the client considers the risk high—though shared—they may try to get the supplier to take on a larger share of it.

Dependence on the client's commitment. If communication is poor or the client doesn't take on their role in the development, the risks increase and both parties are harmed. The client must be fully committed and collaborate directly with the supplier.

CHAPTER III

CREATING AN AGILE CONTRACT

Steps for defining the terms of the contractual framework and choosing a contract model.

In the traditional framework, anything not stated in a contract is handled through a formal change request, defined in the contract itself (Chabes Floreano, 2020). Even where the mechanism exists, changes are seen as something to avoid. In the agile framework, by contrast, change is always a possibility, so we find a wide variety of contracts whose basic elements can be modified and adapted during negotiation. In some cases and with certain contract models, agile contract negotiations may even carry on into the development phase.

This chapter develops general considerations to keep in mind when selecting specific models and clauses.

Defining the contractual framework

Before writing an agile contract, the terms of the contract's framework must be defined, in what is known as a master service agreement: a base agreement for the contract (Fritz & Fischer, 2021, p. 4). This takes into account:

- **The main subject of the contract:** the project's vision.
- **Project governance:** how the project will be led and documented, including roles.
- **Price/remuneration:** the terms for charging for the services.
- **Cooperation with the client:** distinguishing the client's tasks from the supplier's.
- **Service levels (Service Level Matrix):** details of the services the supplier commits to providing, such as quality standards, response times, and penalties if these conditions are not met.

This base agreement is drawn up before the contract is written, in order to surface risks and share responsibilities.

When development relies on artificial intelligence tools, it is worth adding one more consideration to the framework: the permitted use of those tools, ownership of the code they generate, and the handling of the client's data. This is addressed in detail in "Agile contracts in the age of AI" (→ Chapter 5).

Choosing a contract model

Some companies, rather than looking for specific principles and clauses that fit their projects, copy and paste clauses that may not meet their needs. But there is no template contract that can be applied to any situation. Each project will have a contract model that suits its objectives and needs, and that should be determined in advance. So which model to choose?

To decide among the different types of contract, a number of factors must be taken into account:

- The size of the project.

- Command of the technologies needed for the development.
- The degree of maturity and experience with agile methodologies.
- The self-management and self-sufficiency of the supplier's development team.
- The distribution of risks and rewards between the parties.
- Trust between the parties.
- The type of client–supplier relationship: competitive, cooperative, indifferent, or dependent (Hernández, 2019).

There are a number of elements that must be addressed in every agile contract but that may vary depending on the contract model and the factors above. According to Arbogast et al. (2010), these are the elements defined throughout the rest of this chapter.

Common elements

The delivery cycle

Agile models that manage continuous progress through timeboxing deliver a finished part of the product at the end of each iteration. In traditional contracts this is usually known as milestones, set by date or by objectives. However, delivery cycles have some distinctive features:

- **The result of each cycle is deployable.** Each iteration delivery is a finished, tested part of the product, ready to use.
- **The duration is shorter**, usually ranging from one to four weeks.
- **The project's scope is variable**, and before each new cycle, what will be developed in it is usually decided and estimated.

The contract should define aspects such as the average length of the cycles and the conditions that must be met to approve the result of the iteration (what in Scrum is known as the "increment"), although this is a separate element to define (the "acceptance criteria").

Project scope

Agile contracts do not define their scope precisely, though some are more specific than others. There are two extremes: on one hand, contracts such as fixed-cost ones, where the scope is identified at the outset but with mechanisms for change; on the other, contracts such as progressive ones, where there is no need to define any scope beyond the first iteration.

It is inadvisable to write a contract with too vague a scope and vision. If this aspect of the contract becomes so vague that it is inscrutable, it can lead to conflict down the line and to a lack of commitment. If the project risks falling into this trap, the ideal is to let the legal advisors define the scope from their own understanding and to

refine that definition with their help. They can be brought into the project's vision through workshops and other activities.

Change control

Changes to the project's scope are the area that **requires the greatest care when drafting the contract**, so as to make change easy and frequent. It is advisable not to require change control boards, requests, or change processes.

Change concepts in agile project contracts fall into two categories: change in the relationship between the parties, and changes to the project's scope.

Changes in the relationship between the parties

For example, if an acquisition takes place during development. To address this kind of change, the typical format of traditional contracts may be applicable.

Changes to the scope

To make the scope flexible and avoid bureaucratic friction, it is advisable not to require formal meetings or processes to request, document, and approve changes. In an agile project there are other systems for this, and they normally form part of the methodology being used. The Product Owner and the team are usually empowered to make these decisions, to the extent the client sees fit, and changes are decided during the meetings typical of each model, collaborating directly.

It is worth noting that the level of flexibility can range from very high and without penalties, in progressive contracts, to medium, with shared gains and pains, in fixed-price contracts (Arbogast et al., 2010, p. 520).

Termination

This element relates to the agile philosophy that changes are welcome. Although early termination has negative connotations, in an agile project it can be a desirable event: it may mean that success was achieved sooner than expected. And if the collaboration doesn't work out, an easy separation is preferable for both parties.

The ideal termination model in an agile contract is one that lets the client stop at the end of any iteration, without penalty. However, without penalties, the supplier may be left worse off. For this, there are variations on the termination clauses.

Acceptance criteria

Acceptance of the increment produced at the end of each cycle is based on an agreement, a set of previously agreed tests. In Scrum, these are usually referred to as the "Definition of Done."

Sometimes there are changes of mind or rejected deliveries after several iterations, and this can create conflict. That is why it is important to define clear terms for completion, acceptance, and correction.

On the other hand, iterative deliveries and direct communication with the client simplify the acceptance process: over time, the approval and agreements for each deliverable accumulate. Development doesn't happen blindly, but by collaborating and regularly confirming that things are moving in the right direction.

Deliverables

Traditional contracts often include a detailed list of what must be delivered. These details are sometimes incorporated into an appendix of the "quality plan." This format can be too specific and rigid for agile projects, and can bring unwanted consequences, such as:

- Focusing on negotiating and complying with the quality plan instead of cooperating to create useful functionality.
- Reinforcing predictive command-and-control planning instead of the collective capacity to learn and respond to change.
- Reinforcing the belief that a project can be defined in detail, and generating unrealistic expectations of predictable deliveries.

For this reason, agile contracts have other ways of defining deliverables, so that they respond to the philosophy of frequent, incremental value delivery.

At this point, the technical documentation that may accompany each deliverable must also be considered. This documentation, usually specified in detail in traditional contracts, is requested as support for the project's future maintenance and as an aid for new team members. However, maintenance is usually subcontracted to the same people who develop the project, so exhaustive documentation is not necessary. Requiring it as an early deliverable can be a waste of time. One option, if it is a need for the client, is to run a joint agile documentation workshop.

Payment frequency

The most popular and widespread system is to make one payment per iteration, once the deliverable is accepted. The simplest approach is to agree to pay 100% of the agreed price per iteration at the end of each cycle.

Warranties

Concerns about warranties ease incrementally as the project advances. The risk profile associated with the final warranty is lower thanks to the trust built and the acceptance of deliverables.

As with liability limitations, the warranty should be tied to each incremental delivery of work (at the end of each iteration), even if there is a general warranty for the final product.

Techniques such as the aforementioned "Definition of Done" and continuous performance measurement are often used to establish and measure warranties. The first is used to assign general acceptance criteria for marking user stories as done. As for performance measurement, key indicators (KPIs) can be set to measure the supplier's performance and the project's progress.

Liability limitations

Liability limitations are clauses that set a cap on the amount of liability a party will assume in the event of a breach of contract. In agile contracts they can be less specific than in traditional ones.

Negotiating these clauses is usually difficult, but establishing them helps overcome difficulties.

In a traditional contract, these clauses tend to protect the supplier: they limit the amount of damages it is liable to pay in the event of breach, exclude certain types of damages, or require the client to take certain steps to mitigate damages before filing a claim. For example, it can be stipulated that the client cannot recover more than they paid for the goods or services.

In an agile contract, the risks from unforeseen events are eased, since development is progressive and a useful increment is delivered at the end of each cycle. These increments, once used, make it possible to identify problems and make changes before their impact becomes overwhelming. This reduces costs, as well as damage to the client's image and to the relationship between the parties, and it should be taken into account when drafting liability-limitation clauses. It is also worth reflecting that the client must be involved in identifying and correcting faults as they arise.

Pricing

This varies depending on the type of contract. The price can be set by unit of time (per iteration or cycle), by unit of work (usually known as a "story point" or "function point"), by use of the implemented solution, and so on.

Pay-per-use models can make sense in software development projects, where each "use" of a system customized for the client is tracked automatically by the supplier. The client is billed regularly based on how often it is used. This approach aligns the interests of client and supplier, since both parties win if the system is used more and more.

CHAPTER IV

MODELS AND CLAUSES

An inventory of models and clauses for agile contracts, in alphabetical order.

Contract models

Bob Martin

This contract model is the work of Robert Cecil Martin, an engineer and the author of several software design principles. He is also one of the co-authors of the *Agile Manifesto*.

His proposal combines the time-and-materials contract, hourly pricing, and story-point pricing. According to Martin (n.d.):

- **If the actual effort equals the original estimate,** the client's payment is equivalent to a simple time-and-materials contract.
- **If the actual effort varies,** the client's payment varies less severely than in a traditional time-and-materials contract.
- **Client and supplier share the pain/gain** if the project requires more or less effort than the original estimate.

Capacity-based (per-iteration)

This model suits complex projects, where requirements change constantly between sprints and the supplier has to work independently: projects with a high level of development.

Characteristics

According to Hernández (2019):

- The scope is open.
- The Product Owner can introduce all the necessary changes.
- Total project cost = team velocity × number of sprints planned up to the release date × value of the story point.

Things to keep in mind:

- The reference user stories.
- The agreed Definition of Done.
- The Product Owner must be effective at writing acceptance criteria and prioritizing the stories that truly add value.

Interests:

- The supplier must deliver the planned points in order to continue with more iterations.
- The client has predictability of costs and of the amount of functionality they will obtain over time.

Advantages and disadvantages

Among the **advantages** are the high level of agility of this model and the wide room for maneuver and creative freedom it gives the supplier. As a **disadvantage**, an uncertain total cost and negotiations that can take more time than other models.

Cost-reimbursable

This type of contract is used when the scope is uncertain and costs cannot be estimated well enough to make a fixed-price contract.

The contract estimates a certain sum in advance, a cost that both parties consider likely. During the project's development, the supplier is responsible for keeping a record of the actual costs. If they are lower than the agreed sum, the client benefits. If additional costs are incurred, the client must absorb them. It is a type of agreement with more risk for the client.

Other versions

Cost plus incentive fee

The supplier is reimbursed for the agreed costs of carrying out the project plus an additional payment for its fees. These fees are based on pre-established incentives: if the supplier meets the objectives, it receives an additional reward in the form of fees. If the final cost of the project differs from the one estimated at the outset, client and supplier share the variances according to a previously negotiated formula. For example, it can be agreed that 80% of the variances fall to the client and 20% to the supplier.

Cost plus fixed fee

The supplier is reimbursed for the agreed costs of carrying out the contract work plus an additional payment for its fees, calculated as a percentage of the project costs estimated at the outset.

Cost plus award fee

The supplier is reimbursed for all costs, but most of the fees are earned based on the satisfaction of certain general, subjective performance criteria defined and built into the contract (Chabes Floreano, 2020).

Fixed-price

These are the most common in medium- and long-duration projects of great complexity. These contracts clearly define the work requirements and are recommended only when the parties have enough competence and market experience to agree on a realistic provisional price. Although in this respect they are similar to traditional contracts, there are important differences: the price is estimated using agile techniques and is not final until a period of time has passed.

Estimation

To be able to agree on the price before the project is developed, the project backlog—at least the first cycle—is broken down into user stories. The supplier uses reference projects to estimate the time and effort needed to complete each user story in points (story points). Finally, the parties must agree on a fixed price for a given number of story points. The contract should reflect a commitment to hold joint review meetings after each sprint, to discuss whether the scope needs adjusting.

Exit point

The contract must reflect a period of time, from the start of the collaboration to what is usually called the "exit point" or "point of no return," during which the client can back out without penalty and the project price remains alterable. It is a time to check whether the estimates made seem accurate or not. From the "point of no return" onward, the price becomes fixed and the collaboration is consolidated.

Advantages and disadvantages

In addition to the general **advantages** of an agile contract, this model makes planning and resource allocation easier. It can give more confidence to a client unfamiliar with agility, while still being a flexible contract adapted to agile methodologies. As a **disadvantage**, its negotiations can also take time, though it allows development to begin while the price estimate is being refined.

Other versions (Arbogast, 2010)

Fixed Price Incentive Fee

This allows for variances in performance, with financial incentives tied to meeting the agreed metrics. For example, in a consulting-services contract, the supplier can set a fixed price plus an additional fee if it achieves a specific rate of return for the client. The fixed price ensures both parties have a clear idea of the project's total cost, while the incentives provide extra motivation to achieve the desired outcome. It is important that the goals and objectives be clearly set out in the contract to avoid misunderstandings.

Fixed price per iteration (per unit of time)

This is different because the price estimate is simpler and more predictable. It is common among agile subcontractors. There are two ways to approach it:

- **The requirements are defined and agreed before the iteration.** The supplier has to clarify the work and trust the estimates. A problem for clients can be that the supplier adds a contingency fee, due to the risk associated with the variability of research and development work.
- **The requirements are highly flexible or there are no predefined requirements.** The key issue is the client's trust in the supplier. There must be transparency, delivery must be frequent, and it must be easy to end the project.

Fixed price per unit of work (UoW)

Unlike traditional development, an agile UoW reflects the seventh agile principle: working software is the primary measure of progress. These models go by several names: "price per story point," "price per function point," "price per feature point," and so on. The "point" is always related to an estimate of size or effort and, therefore, to cost. That said, some argue that a story point is not an exact measure of value. Generally, a few schemes can be used to determine the fixed price per point: for example, calculating the average based on several previous projects, or a customized amount. In the latter case, the client pays the supplier's average point value over a few iterations; then supplier and client agree on a customized fixed price per point, based on that cost plus some profit margin. What matters in this kind of model is that client and supplier have a clear, shared framework for defining a point[3].

Fixed-price and fixed-scope

This model often comes up in conversations about agile contracts, but it is not recommended.

It is defined by a fixed scope divided into deliveries, working in cycles. Payment is usually made per iteration, with a lump sum at the end (if the budget hasn't run out sooner). The payment per iteration is a fixed percentage of the total price, based on an estimate of the total number of iterations or their duration.

It allows for the possibility of changes, but with added friction. A fixed scope means these are not desirable, and so there are usually formal processes to request and approve them. These are contracts that try to overlap with the idiosyncrasies of an agile environment—such as deliveries in short cycles and requirements as user stories (Stevens, 2019)—but that contradict themselves and lead to problems similar to those of traditional contracts.

Advantages and disadvantages

There are hardly any advantages to be seen. They are sometimes used to give the client greater security and confidence, but this is a false sense of security. Since the scope is fixed, change management is designed to prevent, limit, and hinder any modification of the backlog. But changes in agile projects are neither predictable nor avoidable, and the friction they generate will affect the project's progress and budget.

In these models the iron triangle appears: scope, price, and duration are fixed. The client fixes the scope, and the supplier fixes the cost and the time. Most of the risk falls on the supplier: it must analyze the requirements and the effort needed to meet the client's expectations, and make a better offer than the competition. But if its estimates fail, it will lose money. If the risks and effort have been underestimated, or if too low a cost has

[3] The rise of generative AI strains the assumption of a stable relationship between effort and value on which "per story point" models rest; this is analyzed in "Agile contracts in the age of AI" (→ Chapter 5).

been agreed to, the supplier loses—and in reality the client does too, since the project will suffer.

Many of these limitations revolve around restrictive delivery dates, which tend to matter a great deal in this type of contract. When it is impossible to meet the promised deadlines, the wrong strategies are often used, harming both client and supplier (Hernández, 2019). For example:

- **Overwork:** overtime is unpaid, so there is less motivation and worse performance.
- **Increasing capacity:** new hires are made, which means investing time and effort in explanations. This increases the complexity of communication and coordination, and the potential for conflict. Minimal concessions are made by the client, in an endless cycle of extensions from the supplier.
- **Dropping or altering events.** The daily meeting can turn into a constant argument (especially when the delays are obvious) focused on assigning blame. In extreme cases, Scrum is abandoned entirely in an attempt to meet the date by any means.

The supplier–client relationship can suffer from false expectations and unforeseen events, and can range from competitive to indifferent. When the client asks for more than was agreed, the supplier tends to give in or over-promise to keep them. On top of all this are the conflicting interests that arise from all the parties involved (Hernández, 2019):

- **Crossed interests.** The client tends to exert too much control. Its various stakeholders may want to make changes to suit themselves, so the supplier will have to give in to the pressure to keep them.
- **Interdepartmental objectives are obstacles.** For example, if the sales area handling the bid has unrealistic revenue targets: closing the deal at any cost, even if it later proves unfeasible in terms of time or budget.
- **Against the user.** The supplier will try to deliver a product that meets the requirements on the agreed date, even if many features are defective or not implemented, since it will later have a maintenance contract during which it can resolve these issues without pressure. It is the end user who suffers from the poor quality.

Hybrid

Hybrid contracts in this context may seem redundant, but it is a category worth applying in certain cases and one that is often mentioned. According to Chabes Floreano (2020), they are contracts halfway between the traditional and the agile approach. They combine, for example, several contract models and several types of clauses. They usually appear in large projects and can be achieved by applying agile clauses to contracts that otherwise have a more or less traditional format. For example, a time-and-materials contract with clauses.

Sprint Contract

This is usually used for small projects, completed in a small number of sprints. It isn't really a commercial contract but rather an agreement between the Product Owner and the team. It can be included in the commercial contract, which may be of the time-and-materials type.

Characteristics

- **The scope can vary,** though it is measured after each sprint. The team commits to delivering the agreed requirements to a defined quality standard by the end of the sprint. The Product Owner agrees not to change their instructions before the end of the sprint.
- **There is no possibility of introducing changes** (since they would fall within an ongoing sprint).
- **It is suitable for working in Scrum** on micro-developments or small enhancements, though it isn't a very common use.

For this type of contract, it is common for both supplier and client to know agility and to have been working this way for some time. What does it involve?

1. A team is assigned to a project.
2. If its velocity is unknown, the first three sprints are assessed to obtain it.
3. A price is set per team and sprint, or per story point delivered, assuming the number of points they can complete.
4. The contract ends when the desired value is reached.

Venture capital financing model

This financial model is usually used in venture-capital investments to assess, at an early stage, the viability of investing in a company. It uses long-term financial projections (revenue, expenses, cash flows, valuation, and so on), as well as risks and uncertainties, to determine the potential return. After each work period (between each round of financing), the terms of the contract are reviewed.

Clauses

In an agile contract, clauses are designed to be flexible and to let the parties work together to achieve the contract's objective. Although it can be argued that using them doesn't guarantee adherence to the agile ideal, a conscious use can result in hybrid contracts suited to the project.

Clauses can define certain obligations and responsibilities, such as:

- The objective of the contract.
- The goods or services to be provided, and their limits.
- How communications between the parties will be carried out, including points of contact and protocols.
- How changes to the scope will be managed over the course of the project.
- The payment terms, including price, deadlines, and conditions.
- Each party's responsibilities and penalties in the event of breach.

Bonus/penalty clauses

Bonus and penalty clauses consist of rewarding the supplier for finishing before the agreed date and penalizing it for delays.

Characteristics (Stevens, 2019)

- The amount of the bonus and the penalty usually depends on the deviation from the delivery date.
- This clause makes changes harder to accept, because they always affect the delivery time.
- The risk falls entirely on the supplier. The less time it takes, the more money the client loses, so the client may prefer that there be delays; there must be trust and good communication between the parties.
- The client–supplier relationship can be cooperative, but it can degenerate into indifferent if the agreed date isn't very important to the client.

Changes for Free

In Spanish, "cambios gratis." It is a clause created by Jeff Sutherland that combines with the next one (Money for Nothing). It establishes a mutual agreement when estimating requirements.

Rules

- The Product Owner prioritizes the product backlog before each sprint.
- Changes in priority are free as long as the total amount of work doesn't change.
- New requirements can be added at no cost within the limits of the sprint, if their effort is comparable to that of others already estimated.

- In return, the client gets involved: they follow the project closely and collaborate with the Product Owner to maintain a quality product backlog.

Money for Nothing

In Spanish, "dinero por nada." It is another clause created by Jeff Sutherland. It establishes a mutual agreement when estimating risks.

Characteristics

- The client determines when developing a requirement (or several) costs more than the value it will later provide.
- The client can terminate the contract at any time; if they do, they pay the supplier a percentage (usually around 20%) of the value of the stories that remained to be developed.
- The supplier assumes the risk of delivering late on the work that was mutually estimated and agreed.

Money for Nothing, Changes for Free (combination)

These are two clauses that encourage client and supplier to approach the relationship from the standpoint of value. The way of working is perhaps like a time-and-materials contract with a cost target, usually with the idea that the project shouldn't exhaust the budget. The difference is that there is an incentive for the client (who pays a percentage) and for the supplier (which saves costs and gets money for free) to finish early.

- **Once a certain amount of the project has been delivered**, if enough value has been achieved and no more development is needed, the project is canceled.
- **If the project is canceled before the estimated deadline**, Money for Nothing applies: the client pays the supplier a percentage of the remaining commitment, usually matching the supplier's net profit for the service time not rendered.
- **Changes for Free:** the Product Owner can introduce as many changes as they see fit, always within what Scrum prescribes. If, for example, they introduce changes during a sprint or influence the estimates, it is not possible to continue and it becomes a time-and-materials contract.
- **The scope can change.** It is planned, but it can have changes within requirements or user stories of the same size. Making larger changes or adding new requirements entails extra money.
- **The risk is shared.** It is in both parties' interest to complete the project on time or early: the client has fewer losses, and the supplier a greater margin.
- **If the budget is exceeded,** the rules of the fixed-price contract or of a maximum-cost contract can be applied. To achieve a cooperative relationship, applying the fixed price seems more advisable.

Not to exceed with fixed fee

This involves setting a minimum budget (fixed fee) and a maximum one (not to exceed). In these cases it is usually advisable to reach agreements with the client and, where necessary, to include in clauses what happens if the maximum budget is exceeded.

Trust sprint

This is usually added in contracts with clients who need to feel more secure before entering into a contractual relationship. The trust sprint makes it easy to terminate the contract without penalty after a set number of sprints. It is similar to the Sprint Contract, with the difference that it can be part of any other type of agile contract. It is common for companies that have never worked with agility and for suppliers who want to show the client the advantages of the agile approach.

CHAPTER V

RECOMMENDATIONS

Recommendations drawn from the sources consulted for creating and negotiating agile contracts.

Creating contracts

This section is divided into four points:

- **Finding the best partner/supplier:** advice for the client on choosing partners or suppliers.
- **On legal advice:** advice on the importance of turning to legal advisors and lawyers, not only for drafting the contract.
- **On contracts and clauses:** advice on the clauses and types of contract suited to each kind of project.
- **Creating, presenting, and negotiating a contract:** advice on drafting, selling, and negotiating an agile contract.

Finding the best partner / supplier

For the client, one of the key points before drawing up the contract is identifying a good partner. The project's success depends on the commitment of both parties and on the relationship being cooperative. Contractual requirements should be selected to ensure an optimal balance of cost, time, and quality for both parties.

Sometimes another external supplier, or an internal resource with sufficient knowledge, can be tasked with playing an operational role in implementing the project. According to Fritz and Fischer (2021), this external service can act as Scrum Master or Product Owner.

This decision should not be taken lightly. The Product Owner is a key strategic position for the project's success, and it is generally recommended that this be someone from the client's company, who knows the business and the vision inside out. That said, if the right person is found, a good consulting service can add value (Léonard, 2019). It can be a good idea if the organization doesn't have enough experience or doesn't have someone who can devote themselves 100% to the Product Owner role.

On legal advice

For Arbogast et al. (2010), it is important that the advisor or lawyer who is going to draft an agile contract has information about agility or has researched it. A lawyer with traditional training will tend to distrust models and clauses they don't know or don't consider proven enough, in order to protect the client.

Legal advisors without prior experience in agile projects may start from mistaken assumptions. For example, being unfamiliar with the uncertainties and variables typical of a software project, they may approach it as a staged project, like a construction one. These assumptions don't fit agile development (Arbogast et al., 2010, p. 502) and are very likely to make their way into the contract's language, harming the parties.

In addition, the traditional view of legal advice also shapes the definition of a project's "success." Factors such as whether end users are satisfied with the product, or

whether the product can be a success even if its initial requirements have changed, are usually not taken into account (Arbogast et al., 2010).

On the other hand, lawyers often say that if both parties are somewhat dissatisfied, the contract is a success: both have had to give something up to reach agreement. But this mindset can contradict what is sought in agility, which is win-win situations.

On contracts and clauses

Fritz and Fischer (2021) recommend different contracts depending on the complexity and duration of the project:

- **Simple, short-duration projects, focused on costs and with a clear vision.** The authors recommend what they call a "work contract for the holistic project," pointing to transparent, predictable costs. However, this model has a low degree of flexibility. Instead, the Sprint Contract might be more advisable; or the capacity-based (per-iteration) model, if the project isn't quite so small.
- **Complex, long-duration projects.** They recommend a capacity-based (per-iteration) contract, which allows adjustments to be made on the fly rather than trying to plan the unpredictable from the start. Bear in mind that in this case costs are uncertain, since there will be adjustments from one iteration to the next. Agile fixed-price contracts can also be used, with the disadvantage that they require more planning time and a trial period.
- **Projects executed and controlled by the client.** Here the authors talk about service contracts. Since the supplier isn't bound by the project's success, the internal resources must have enough experience in managing agile projects to guarantee a certain standard of quality and profitability.

As for clauses, Hernández (2019) considers that using the Money for Nothing and Changes for Free clauses in projects or micro-developments of 200 hours is not very productive.

Creating, presenting, and negotiating

Before we even consider drafting a contract, we must put in place a framework for agile contracts—in other words, prepare the ground. Hernández (2019) recommends the following:

- **Look into the market** to find out which suppliers offer agile approaches.
- **Train the Product Owner.**
- **Run trials:** you have to learn to be agile, and for that you need to practice. For example:
    - **Take advantage of small enhancements or micro-developments** of under a month to run a single sprint as a trial for learning about agility.

    - **Take advantage of small pr**ojects (2 or 3 months' estimate) using time-and-materials contracts to set up a Scrum team and learn its velocity.

Suppose we already have some experience and have done projects with an agile methodology, and now we want to apply it to a larger project. For Hernández (2019), we must be clear on several points:

- **Each contract responds to a need:** using the Stacey matrix can be helpful, because it proposes different ways of developing depending on the degree of certainty in the requirements and command of the technology involved.
- **The scope should not be fixed** if the size and complexity of the project are decisive.
- **Dysfunctional Scrum implementation:** if the Scrum implementation is dysfunctional, or we started applying it barely a week ago, an agile contract will not solve every problem.

Once these points are clear, you have to choose the type of contract and the clauses. There are different opinions on the best way to draft a contract or which elements to avoid and include. Nonetheless, one common view is the one shared by Tortorella (2012) in his video Kleer — *Contratos ágiles en la vida diaria*: the first agile contract should be as simple as possible (price per sprint, per person, per hour, and so on), though it should include situations that could arise. It won't be perfect and will have to be modified. As many clauses as necessary can be added, suited to each project and client.

If you know the agile approach and are working with a client who also knows it, Hernández (2019) proposes making framework agreements with a price per story point. And if we already have a Scrum team whose velocity is known, it can be assigned to a larger project using a capacity-based contract or a time-and-materials one with a cost ceiling and variable scope.

For Arbogast et al. (2010), what is advisable when drafting a contract is:

- **Avoid incentives and penalties.** Incentives and bonuses are assumed to be beneficial, but some consider that they reduce transparency and quality and increase competitiveness between client and supplier. The same happens with penalties. An alternative can be share-the-pain/gain models, which foster collaboration and joint improvement.
- **Avoid the Quality Management Plan and the Deliverables List.** Traditional contracts usually imply low levels of trust and transparency. One of the traditional solutions is to require a conventional "quality management plan" or a "deliverables list" defining a long documentation checklist, instead of focusing on delivering real value. In Scrum there is a Definition of Done, produced by the supplier's team and the client, that is worked out with each iteration: it is based on practices and artifacts, is redefined at the start of each iteration, and evolves according to what each party considers adds value.

- **Collaborate with lawyers early and often.** It is important to turn to legal advisors before drafting any contract and during the project's development.

To draft a contract, it is also relevant to know how to propose it to the client and negotiate. For Tortorella (2012), the proposal of the contract and the project depends on the type of client. He also considers several of the principles we discussed in Chapter 3 to be fundamental: transparency, trust, and communication.

To give the client confidence, a pilot project or a first sprint can be offered that entails no expense if rejected. In fact, it can be included as a clause, and the project may not carry much risk.

Transparency and communication also play an important role in "selling" and negotiating the contract. On one hand, the client must be clear that estimates are just that—estimates. Saying that a task may take two weeks is not a verdict, it is a possibility. This also helps manage expectations, an important point that must be made clear in every department, especially in sales: promise what you can deliver. Salespeople have to learn to say no (Tortorella, 2012).

On the other hand, the client must know what is being done. To this end, Tortorella (2012) proposes using what is known as an agile engagement road map: showing how the iterations are handled and explaining to the client in how many the project can be done. That said, before reaching this point and after signing the contract, three prior stages must be taken into account:

- **Check the fit:** establish business relationships.
- **Project inception:** explore and assess.
- **Iteration 0:** plan the project's infrastructure.

Negotiation tools and strategies

The strategies and tools explained in this section are drawn from the book *Agile Contracts. Creating and Managing Successful Projects with Scrum* (2013), by Opelt, Gloger, Pfarl, and Mittermayr, which develops in detail how to negotiate and bid for agile fixed-price contracts. Some general advice for negotiating agile contracts can be drawn from it—mostly suggestions for convincing clients unfamiliar with agile methodologies.

Negotiation strategies

The goal of the negotiations is the same for both parties: to negotiate and implement a successful project. However, each party has different interests that influence the negotiation:

- **The client's objectives:** commercial and other significant objectives, which will depend on the type of business and contract. Ideally, they are defined from the start and shared with the decision-makers, so that a clear line of negotiation can be followed. Once specified, the implementation method must be decided.
- **The supplier's objectives:** also commercial and significant. A significant objective might be to include certain basic implementation principles in the contract; a commercial one, how to negotiate the bonus system. Both vary depending on the type of contract, but also on the market situation: a supplier may not yet have recognition in the industry and may decide to offer very aggressive prices to gain market share.
- **The objectives of the team and the people involved:** related to the bonus payments of the people involved on both sides. They are called financial or non-financial, tangible or intangible incentives. Again, they vary depending on the type of contract and project.

A first step, therefore, is **to define the principles for the project and the negotiation from the outset**. If an agile approach is chosen, both client and supplier must agree on how they will conduct the negotiation cooperatively, following that approach.

As suppliers, we must bear in mind that **choosing the right supplier matters to the client.** The client will ask what service and what success they expect from a supplier; that is, they will use strategies to find out which suppliers can guarantee their objectives. That is why it is important to attend to two aspects: the negotiation team, and the agenda and documentation.

The negotiation team

The negotiation team and all stakeholders must be discreet and keep the project's objectives, negotiation strategies, and tactics confidential. It is common for negotiating partners to try to obtain information, for example by holding conversations with employees.

Opelt et al. (2013) propose using the FBI's organizational model: the negotiation is carried out by a team with three strictly assigned roles.

1. Negotiation leader

A single person. They focus on the negotiation, try to identify possible misunderstandings, and seek to optimize all the relevant aspects of the project. In an initial phase, their role is to test the negotiating partner's claims and secure information. In principle, they never decide alone on essential matters: they must coordinate with their team on all decisions outside their remit, which allows for more time and avoids rushed decisions. One of their greatest challenges is the **emotional task**:

- Highlight the strengths of one's own technical team and put the supplier's into perspective, in order to get the other party to accept a cooperative contract model.
- Provoke emotional mistakes in the negotiating partners, without being provoked oneself.
- Be firm with one's demands, but make real, significant concessions to the supplier.
- Don't offer compromises, but obtain compromise proposals on any topic.
- Project a convincing position of power and not be intimidated.

2. Commander

They support the negotiation leader. They don't intervene directly, but follow the negotiation silently and provide guidance during breaks. However, if this isn't enough, the commander can end up intervening. They also act as a liaison between the decision maker, whom they keep informed, and the leader, who must focus on managing the negotiation.

3. Decision maker

The one who makes the decisions. Usually an executive of the organization, who stays out of the way so as not to be swayed emotionally. They must know the project's requirements and have no personal preference for one supplier over another.

Agenda

You must make sure that, once the contract is finalized, as few matters as possible remain open. The initial draft of the contract should contain some elements that can be reconsidered in the negotiation. In general, you should aim to set the agenda—and therefore the content of the negotiation—yourself.

The goal is a balanced contract that fosters cooperation and, consequently, ensures the project's success. With agile models, negotiations should not be considered complete until the project has been successfully implemented: because of the possibility of changes, it is normal to have to review the contract or the agreements during development. That is why the ideal is to make a basic set of rules clear for each project, before and after the price and contract negotiations.

Negotiation tools

The tools or strategies for negotiating an agile contract depend on the company you work with, the effort, and the approach. Here are some:

Laying the groundwork

Stimulating interest before the negotiation

The priority is for all parties to understand the general advantages of agile methods and how an agile contract supports them. If a company already works with agile methods, or has at least experimented with them, it can draw on that experience. However, there is often reluctance from middle management (project management), who have to change procedures and ways of working practiced for years. For this reason, the transformation process must include reflection on what each member of each party stands to lose, and demonstrate the individual benefits. To make the case for the effectiveness of this kind of contract, the following arguments can be used:

- **What is really needed gets implemented, not just what is specified.** The logical thing is for the client to express what they want to achieve and for the supplier to make it possible. The client shouldn't focus on the "how" but on the "what"; that way the supplier has the freedom to decide how, and the opportunity to implement the "what."
- **Striking statistics.** The detailed scope of a two-year project usually changes by more than half. The logical thing is to be flexible and try a form of contract that drastically minimizes the risk of stranded investments.
- **Bonus plans.** Discussing current incentive schemes and their real impact on the project or budget is essential to having a motivated, efficient team.
- **Transparency.** With the agile model, you see what is completed and what is not, every two weeks. In the traditional model, progress isn't visible until the end; the same goes for testing, which usually isn't done until the last third of the project, when finding errors means the information arrives too late.

Exchanging experiences

This is about identifying problems from experience in order to try something new. An ideal starting point is when the client is already familiar with the problems of traditional contracts. According to Opelt et al. (2013), most clients bring with them a "base of problems." It is worth taking advantage of this to hold an open conversation from which an exchange of experiences emerges; from those problems and the benefits presented, the client will realize on their own that the shift to an agile approach is advantageous. One option at this point is to offer a trial project with the client's cooperation, to give them confidence and start setting the right expectations.

Establishing a common language and shared experiences

Once there is a shared awareness of the problem, you have to work on the foundations of something new. Mutual understanding requires a common language and a shared vision. It is advisable to organize two-hour workshops, for example. In the first 45 minutes, the basic idea of agile methods is described and realistic expectations are set. Then the group works with everyday examples to "practice" the agile method: for instance, estimating the costs of a three-week trip to the Bahamas using Planning Poker, and then "implementing" it with some other activity, such as the Ball-Point Game.

Discussing the form

Feature shoot-out

This consists of inviting clients to a "feature contest" where the benefits of the various types of contract are compared. Visual aids (flip charts and colors) and images can be used to illustrate the results. It serves to convince a wider circle of the client's employees and to soften the most stubborn objections.

The black swan scenario

This concept appeared in 2011, in an article by Flyvbjerg and Budzier, referring to projects that, through their failure, devoured huge sums of budget and could end careers and even destroy companies (Flyvbjerg & Budzier, 2011). It has been used as a central argument for raising awareness of the need for change. It should be presented to the client and agreed upon through a risk analysis of specific projects; this can help them consider agile contracts, especially if the errors found have a large impact. With the agile approach, transparency and added value are maximized while risks are minimized.

Workshop on contract setup

To show the efficiency of agile contracts, a one-day workshop with the client can be held. The ideal is to work out with them an example based on an agile contract (contrasted, if desired, with a traditional one) and with a suitable context. The point is to put the explained theory into practice and for the supplier to demonstrate how it works and how it handles change.

Reports and metrics

One of the key aspects of the agile model is that the client is always informed about the state of the project. These are not status reports, but a development methodology that makes it possible to measure progress in small binary units (done or not done). On this basis, the reports to be presented periodically are agreed in the contract.

KISS Backlog View

This follows the KISS motto (keep it simple, stupid). Backlog items are assigned to the planned sprint, and a "conversion rate" from story points to team costs is stipulated in

the contract. By building a table, the supplier can see at any time which features have been delivered and which the client has accepted; thanks to the story points, costs can be forecast. If desired, two columns can be added showing which parts have finally been described (in detailed user stories) and which specifications have been approved by both parties. The sprint velocity and the days invested, the people involved, or the team costs should also be shown.

Team metric

This is a way of measuring how the team has been working: it shows the client to what extent it has worked on tasks related to the project and what it took part in, for example product improvements.

Focus: There Is a Single Goal!

In some projects a whole series of objectives is pursued, which inevitably causes a drop in efficiency and conflicts of interest. One way to resolve this is **to set a common goal:** it must have the greatest commercial value and be crucial to the project's success. This doesn't mean that this single goal defines the contract in its entirety or in detail; with such a simple goal formulated, the various levels of project management can make decisions.

Agile contracts in the age of AI

Generative AI is now an everyday tool: according to Stack Overflow's 2025 survey, 84% of developers use or plan to use AI assistants, up from 76% the year before (Stack Overflow, 2025). This surge affects several of the assumptions on which an agile contract rests, and it is worth approaching it with the same critical attitude as the rest of this book: without exaggeration and with the available evidence.

What the evidence says (and what it doesn't)

It is tempting to assume that AI multiplies productivity and that, therefore, estimates and prices would have to be redone. There are nuances, however. Controlled experiments on scoped tasks show notable speedups—up to 56% faster when completing a specific task with GitHub Copilot (Peng et al., 2023), and around 20% in a Google trial (Paradkar et al., 2024). But those individual gains don't translate straightforwardly into the team's delivery velocity: the time saved writing code reappears in review, testing, and integration. The most rigorous controlled trial to date found that experienced developers, working on mature codebases, were on average 19% slower with AI, despite being convinced they had gone faster (Becker et al., 2025).

From this a direct lesson for contracting emerges: team velocity is still something you measure, not something you assume. A well-designed agile contract already handles this (velocity is observed over the first sprints and the price is adjusted accordingly), and that logic doesn't change with AI. What does change is that initial estimates are even less reliable at the start, because the real impact of the tools depends on the type of task, the maturity of the code, and the team's experience. The recommendation to set a "point of no return" after which the price is consolidated (→ "Fixed-price," Chapter 4) makes, if anything, even more sense.

Estimation, story points, and price

Models that set the price "per story point" or "per unit of work" rest on a stable relationship between effort and value. If AI alters productivity unevenly across tasks, that relationship becomes noisier. It doesn't invalidate the model, but it advises:

- **Reviewing the "conversion rate"** from points to cost more often, especially at the start of the collaboration.
- **Not passing on to the client an unverified "AI dividend."** Promising faster deliveries "because we use AI" repeats the over-selling mistake this book criticizes in fixed-price-and-fixed-scope contracts.
- **Watching for the bottleneck to shift toward review and quality**, which can make apparently faster iterations more expensive.

Intellectual property of AI-generated code

This is probably the point that most affects the letter of the contract, and where it matters most to distinguish between jurisdictions. The good news is that, despite the absence of a globally harmonized rule, legal systems **converge on one principle**: copyright requires **human authorship**. What is generated entirely by an AI, without significant human creative input, **is not protected**; what is AI-assisted, with sufficient human contribution, can be. The Berne Convention is the common backdrop to this position.

- **Spain.** The consolidated text of the Intellectual Property Law (Royal Legislative Decree 1/1996) reserves the status of author for a natural person (art. 5). The Registry has denied registration of works without identifiable human authorship, such as the novel Iris, presented in Madrid in 2023 (DiG Abogados, 2025).
- **European Union.** Directive 2001/29/EC and the case law of the CJEU require the work to be the "author's own intellectual creation" (the Infopaq case), which presupposes a human author. The European Parliament reaffirmed this principle in its report on copyright and generative AI (European Parliament, 2026).
- **United States.** The Copyright Office (Copyright and Artificial Intelligence, Part 2, 2025) holds that prompts alone are not enough to attribute authorship. The courts confirmed this: in Thaler v. Perlmutter (D.C. Cir., 2025) it was held that a work generated autonomously by AI is not registrable, and in March 2026 the Supreme Court declined to review the case, so the human-authorship requirement is settled.
- **Latin America.** The pattern repeats. In Mexico, the Supreme Court of Justice of the Nation ruled in 2025 that works generated exclusively by AI are not registrable, since authorship is a right of natural persons (arts. 12 and 18 of the Federal Copyright Law), in line with the Berne Convention and the USMCA (SCJN, 2025). Argentina (Law 11,723) and Colombia uphold the same human-authorship requirement, still without specific rules for AI (Ciencia Latina, 2025).

This convergence is not universal: some Asian courts (in China, for example) have protected AI-generated creations where there was human intervention through prompt engineering. And the framework is evolving quickly, so any contract with an international dimension should be reviewed with local legal advice.

The practical consequence is uncomfortable but clear: **a classic assignment clause of the type "all work product belongs to the client" may be assigning rights over material that has no legal protection.** On top of this is the risk on the training side: an assistant may reproduce fragments of code carrying incompatible licenses from its training data. This is already litigated territory[4] and with real financial consequences: in 2025, Anthropic agreed to pay 1.5 billion dollars for having trained on pirated books, in a ruling that distinguished training on lawfully acquired copies (permissible) from training

[4] In *Doe v. GitHub* (2022), most of the claims were dismissed, and only those relating to open-source licenses and breach of contract remain, on appeal before the Ninth Circuit.

on pirated material (not permissible). It is also worth reading the fine print of AI providers: much of their terms of use shift liability to the user, though some now offer indemnity commitments for copyright claims.

Recommendations for the contract:

- **Distinguish in the clauses** between "AI-generated" and "AI-assisted" deliverables, and assign ownership accordingly.
- **Require a chain of custody of human authorship** (review, substantial modification, documentation of the process) to support the protection of the deliverables.
- **Data and confidentiality clauses:** what client information may be entered into third-party tools and under what guarantees.
- **Third-party license indemnity clause:** allocate liability if the code incorporates fragments carrying incompatible licenses.

This section is informational and does not constitute legal advice; since the framework is evolving quickly, it is advisable to check each contract with legal counsel in the applicable jurisdiction.

Liability and quality

The evidence also warns about quality: AI-assisted code can introduce subtle defects and security problems that push work downstream (Butler et al., 2025). This connects with the warranty and liability-limitation clauses already discussed: the Definition of Done should explicitly incorporate the review of AI contributions, and warranties should address who answers for a defect originating in an automated suggestion.

Regulatory compliance: the EU AI Act

For European clients and suppliers, contracting cannot ignore the EU AI Act (Regulation 2024/1689), in force since 1 August 2024 and applied in stages: prohibited practices and AI literacy apply from 2 February 2025; the obligations for general-purpose AI (GPAI) models and governance, from 2 August 2025; and most obligations for high-risk systems, along with the penalty regime, from 2 August 2026 (European Commission, 2024). Penalties reach up to €35 million or 7% of worldwide turnover in the most serious cases.

The practical implication is clear: if the contracted product uses AI in a high-risk domain (employment, credit, education, infrastructure, and so on) or integrates a GPAI model, **the obligations of transparency, documentation, and copyright compliance must be allocated in the contract.**

Fit with the agile principles

None of the above contradicts the *Manifesto*; rather, it strains it. AI reinforces frequent delivery, but it tests the principle that "working software is the primary measure of

progress" when apparent speed hides technical debt. And it brings back to the center two intangibles this book keeps repeating: transparency (declaring how and where AI is used) and trust (jointly assuming an uncertainty that the tools don't eliminate, only displace). *The 18th State of Agile Report* sums it up well: AI adoption is running well ahead of its governance (Digital.ai, 2025). **The agile contract is, precisely, one of the places where that governance can begin to be written.**

CONCLUSION

In our journey through the terrain of the agile legal framework, we have been able to get a little closer to the concept of the "agile contract." As with project management, agile contracts try to fit into a reality shaped by traditional models, transforming their foundations into agile principles and modifying the elements of the contract, always in pursuit of change rather than its limitation. It could be said, then, that they are hybrid contracts and that they respond to a need for change.

Drafting one depends on the project and on the circumstances and context in which that project was conceived. In other words, there is no template agile contract that is useful and effective for any project, because it is subject to the type of client, the company's situation, the relationship between client and supplier, and so on. Even so, information can be found about the various types of contract that serves as a guide for designing one's own.

As for negotiating an agile contract, we have learned that the strategies and tools are similar to those of a traditional one; the only difference is the nature of the contract. Not every client will feel comfortable working under an agile approach, so in most cases negotiations come down to establishing clear objectives between the parties and convincing the client of the benefits of working with an agile methodology.

In short, agile contracts can be a challenge, in their conception as well as in their drafting and negotiation. Their biggest disadvantage is that they do not yet command the confidence a traditional contract can inspire.

To all of this is added, since 2023, a factor that none of the original sources had to consider: the rise of generative AI. It doesn't change the nature of the agile contract, but it does strain some of its assumptions—the relationship between effort and value, the ownership of what is delivered, the allocation of responsibilities—and it adds new regulatory obligations. Far from making the agile philosophy less relevant, it reinforces its two intangible pillars: transparency and trust. A good agile contract remains, above all, an agreement to change well.

APPENDICES

Information, resources, and professional community

Agile Club →

Skill Arena →

Academy →

BoK/Wiki →

Community →

Comic strips →

About the Scrum Manager® certification →

Frequently asked questions about Scrum Manager® courses and exams →

Social and professional networks

LinkedIn

Youtube

Continuous improvement and quality control

Scrum Manager's quality control is based on the ratings of its students. With your feedback, you help us maintain the standard of our materials, courses, centers, and instructors.

If you have taken part in a training activity audited by Scrum Manager®, we kindly ask—and are grateful—that you rate the quality of the training you received. The information collected is anonymous. You can send us your feedback from the "Members' Area": https://scrummanager.com

References

Books and academic articles

Arbogast, T., Larman, C., & Vodde, B. (2010). *Practices for Scaling Lean & Agile Development: Large, Multisite, & Offshore Product Development with Large-Scale Scrum*. Addison-Wesley.

Becker, J. et al. (2025). *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*. METR.

Butler, S. et al. (2025). *Echoes of AI: Investigating the Downstream Effects of AI Assistants on Software Maintainability*. arXiv:2507.00788.

Eveleens, J. L., & Verhoef, C. (2010). "The Rise and Fall of the Chaos Report Figures." *IEEE Software*, 27(1), 30–36.

Gerster, D., & Dremel, C. (2019). "Agile Contracts: Learning from an autonomous driving sourcing project." *Proceedings of the 27th European Conference on Information Systems (ECIS)*, Stockholm and Uppsala.

Lewicki, R. J., Saunders, D. M., & Barry, B. (2009). *Essentials of Negotiation*. McGraw-Hill Higher Education.

Opelt, A., Gloger, B., Pfarl, W., & Mittermayr, R. (2013). *Agile Contracts. Creating and Managing Successful Projects with Scrum*. Wiley.

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot*. arXiv:2302.06590.

Pichler, R. (2012). *Agile Product Management with Scrum*. Addison-Wesley.

Pries-Heje, L., & Pries-Heje, J. (2014). "Agile Contracts: Designing an Agile Team Selection Guideline." *IRIS*, no. 5.

Takeuchi, H., & Nonaka, I. (1986). "The New New Product Development Game." *Harvard Business Review*, 64(1), 137–146.

Reports and studies

BearingPoint (2020). *Agile Pulse 2020*.

BearingPoint (2022). *Agile Pulse 2022. Doing Agile vs. Being Agile*.

Digital.ai (2025). *18th Annual State of Agile Report*.

Fritz, W., & Fischer, T. (2021). *The Art of Agile Contracting*. BearingPoint.

Stack Overflow (2025). *Developer Survey 2025*.

U.S. Copyright Office (2025). *Copyright and Artificial Intelligence, Part 2: Copyrightability*.

Articles and web resources

Alfonso, A. (2018). "Make Agile Contracts Work For Your Teams & Clients." *The Digital Project Manager*.

Bañeres, J. P. (2012). "Ejemplo de plantilla para documentar el uso de scrum en un proyecto." *Navegapolis*.

Barato, J. (2017). "Contratos ágiles." *PMPeople*.

Hernández, J. (2019). "Contratos ágiles — Primera, segunda y tercera parte." *Agile Experience*.

Kelly, A. (2011). "Agile Contracts." *InfoQ*.

Léonard, N. (2019). "¿Podría un Product Owner ser un consultor externo?" *LinkedIn*.

Martin, R. C. (n.d.). *Fixed Price / Fixed Scope Contracts*. (The "Bob Martin" model.)

Stevens, P. (2019). "10 Agile Contracts." *Agile Software Development*.

Sutherland, J. (n.d.). *Agile Contracts*. Scrum Inc.

VersionOne / Digital.ai (2012). "Must-Haves for Agile Contracts".

Regulatory framework and case law

European Commission (2024). *Regulation (EU) 2024/1689 laying down harmonized rules on artificial intelligence (Artificial Intelligence Act)*. Official Journal of the European Union.

Directive 2001/29/EC of the European Parliament and of the Council on the harmonization of certain aspects of copyright and related rights in the information society.

Spain. *Royal Legislative Decree 1/1996, consolidated text of the Intellectual Property Law (TRLPI)*.

Mexico. *Federal Copyright Law* (arts. 12 and 18); Supreme Court of Justice of the Nation (SCJN, 2025), ruling on AI-generated works.

Argentina. *Law 11,723 on Intellectual Property*.

European Parliament (2026). *Report on copyright and generative artificial intelligence: opportunities and challenges* (A10-0019/2026).

Thaler v. Perlmutter, 130 F.4th 1039 (D.C. Cir. 2025); certiorari denied by the U.S. Supreme Court (March 2026).

Doe v. GitHub, Inc., No. 3:22-cv-06823 (N.D. Cal. 2022)—most claims dismissed; the rest on appeal before the Ninth Circuit.

Videos

Abad Londoño, J. H. (2017). *Hablemos de contratos ágiles* [video]. YouTube.

Ágiles – Liderazgo y cultura (2022). *Contratos ágiles* [video]. YouTube.

Chabes Floreano, J. (2020). *Los contratos en el mundo AGILE* [video]. YouTube.

Schwartz, F. (2017). *Contratos ágiles* [playlist]. YouTube.

Tortorella, P. (2012). *Kleer — Contratos ágiles en la vida diaria* [video]. YouTube.