Contracting agile developments for the Public Sector: the Italian case

Daniel Russo[1], Paolo Ciancarini[1], and Gerolamo Taccogna[2]

[1] University of Bologna, Dept. of Computer Science
[2] University of Genoa, Dept. of Law

Abstract. Even if Agile is a well established software development paradigm, major concerns rise when it comes to contracting issues. *How* to contractualize the Agile production of software, especially for mission & security critical public organizations, is a major concern. In literature, little has been done, from a foundational point of view regarding the formalization of such contracts. Especially when the development is outsourced to a different organization, the management of the contractual life is difficult. This happens because the interests of the two parties are not aligned. Software houses strive for the minimization of the effort, while the customer expects high quality artifacts. This structural asymmetry can hardly be overcome with traditional "Waterfall" contracts. We propose a foundational approach to the law & economics of Agile contracts. Moreover, we explore the key elements of the Italian procurement law and outline a suitable solution to merge some basic legal constraints with Agile requirements. This is a first framework to start building Agile contracts for the Italian public sector.

Keywords: Software Engineering, Agile, Agile Contracts, Contracting, Public Sector

1 Introduction

We study the enactment of Agile software development methods within mission & security critical public organizations [5,11,19]. Apparently, a Waterfall process model responds to some fundamental needs in such organizations, like (i) fixed costs, (ii) requirement definition, (iii) defined schedule, and (iv) clear liability if something goes wrong. However, when costs rise exponentially during maintenance due to poor software quality of the deliverables or the loose requirement implementation, Waterfall shows all his limits. Agile tackles those issues, trying to align the interests of the development team and the customer. However, it is still difficult to formalize a contract ruling agile developments of mission critical products. From a scholarly point of view few similar works have been carried out, especially in the mission critical domain for the public sector.

In the last years some papers have discussed the problem of Agile contracting in a business context, see for instance [3,15]. A. Cockburn has published an intriguing case by case discussion in this site[3].

[3] http://alistair.cockburn.us/Agile+contracts

The US govern has also devoted some attention to the problem of contracts for agile developments. The Software Engineering Institute (SEI) published in the last five years several reports concerning agile for software products for the military [9,16,8,13,14,22].

In particular, SEI has also published some guidelines for agile contracts for software acquired by the US govern [21,10]. These guidelines compare traditional developments with agile developments for critical military systems. The major recommendation consists of post-award documenting contractor's performance throughout each sprint and release, e.g. using metrics like technical debt in terms of bug defect rates, length of throughput time compared to contractor estimates, speed of time to value, etc.

The rationale of this paper is to bind contract theory with Agile practices, wich special care for the Italian context. The foundational approach highlights the key issues of Agile contracting which need to be developed.

The paper has the following structure. In section 2 Law & Economics of contract theory is briefly explained to understand the underling logic of software contracts. This interdisciplinary approach is crucial to understand the economics of contracts i.e., alignment of interests, which is the most tricky part of Agile contracting. In section 3 we deepen the Italian case, defining the key elements of the procurement law. After gaining a short understanding about the basic legal boundaries for Agile public contracting, we illustrate two working solutions. The first one is based on Function Point Analysis in section 4; the second one is based on the contractualization of Scrum sprints, in section 5. Finally, in section 6 we sum up our main proposal and envision some further work.

2 The Law & Economics of Agile contracts

Contracts are agreements between two parties, with different interests, written down to fix such interests, alongside with results compensation. Generally speaking, for a free-market economy, the ability of two parties to enter into voluntary agreements, namely contracts, is the key element for the market equilibrium [7]. Contract law and law enforcement procedures are fundamental for the efficiency of any economic system. Thus, contract law has to be intended as a set of rules for exchanging individual claims to entitlements (i.e., interests). In this way, it enforces the extent to which society gains from this agreement due to an efficient economic system.

When one party is unsure about the other's party behavior, contracts may mitigate this asymmetry, In our case, contracts are helpful when advance commitment enhances the value of an artifact by enabling reliance by the beneficiary [17]

From a Law & Economics viewpoint, there are several issues regarding the importance of contracting [7].

- *Coordination*. The most common reason to engage in a contract is to coordinate independent actions in a situation of multiple equilibria. The most straightforward example is the well known Prisoner's dilemma. Two parties with different and independent interests will choose the scenario where both are worse off (i.e., both confess their crime and accuse the other party, in order to get the benefit of a reduction of imprisonment time). While, with coordination, both would get the better payoff, not admitting the crime, gambling the law system, escaping a long imprisonment time. When the parties are well coordinated by a contract, they will get both

the best trade-off, not going to jail at all. A contract to play this efficient equilibrium guarantees a positive outcome. This is also known as Nash equilibrium, where modern contracting theories get most of their inspiration. The coordination scenarios based on contracts are the best models through which understand institutions [12].
– *Exchange implementation*. Especially in situations of hidden informations (i.e., information asymmetry), contracts may mitigate such asymmetry [1]. To avoid adverse selection (i.e., when one party has an information which the other party does not have), which impedes market efficiency, contracts may provide warranties, to assuring the high quality of the product. This is very typical in software, where the vendors know the details of the product, while the customer is totally unaware of the code (usually obfuscated, if it is a license product) but only aware about its functionalities told by the vendor. Thus, alongside with software, there is usually a warranty about the product. In this way the customer potential downsize (bugs) will be fixed by the vendor and no special code awareness is needed before buying the software.

However, there are also some major drawbacks of contracting [7]. The most important from our point of view are:

– *Specification cost - ex post*. Writing down, foreign all possible contingencies which could arise within the future contractual relationship is extremely expensive. Potential contingencies of contractual obligations are usually very broad. Therefore contracts are often left open and incomplete. In such cases there are two main scenarios. It could happen that the contract just fails to provide information for contingencies, since nothing was agreed upfront. In this case, parties have to decide what happens after a contingency. In the second case the contract could cover a broad number of contingencies but not fine-tuning them. In such way, parties still have to decide what to do do after, since contingencies are not defined precisely enough. Anyway, in both scenarios, contracts fail to assure the commitment of the parties.
– *Dynamic inconsistency - ex ante*. This is the classic investment problem. One party may be willing to bargain and to modify the contracts when has pursued investments. A software vendor will try to sell its solution to a higher cost, if it realizes that the value added brought to the customer is higher than expected. In such case, vendors do not have any incentives to do investments i.e., spend money to develop high quality code, since the price has been fixed.
– *Unverifiable actions*. Even after entering into a contractual commitment, one party may be unable to determine whatever the agreement has been kept or broken. This is the typical case of intangible goods, like software. It is a not trivial task to asses with objectivity if what promised has been carried out according to the contract.

In the study of Agile contracting we should not overlook normative and incentive aspects, typical of any contractual relationship. The economics of contracting has both upsides (i.e., coordination and exchange implementation) and downsizes (i.e., specification cost, dynamic inconsistency and unverifiable actions). What we learn from Law & Economics of contract is that any contract has its loopholes, even Waterfall ones.

Both specification costs and unverifiable actions have a big impact on cost of contracting. Traditional software contracts are very expensive, alongside with high specification costs due to very detailed requirements, there is also the difficulty to assess with objectivity the artifact. Such barriers have a direct impact on both contract cost and market efficiency. Even in Waterfall contracts there are "hidden" costs that indirectly increase the cost of software products. The perceived "reliability" of Waterfall has apparently loose evidence in practice. What we do know is that Waterfall increases incredibly maintenance costs, which are hidden costs belonging to the software's life cycle [18]. However, there are established routines how to carry out a Waterfall contract, instead there are very few about Agile.

First of all we will depict divergent interests of a software contract, represented in Table 1. As seen before, contracts facilitates market equilibrium through coordination and exchange implementation. In software this means that two parties which suffers from a information asymmetry reaches an agreement through a legal binding paper (the contract). The organization do not always have the expertise or the man-power to carry out the software, while contractors do. Both parties are not aware of relevant informations, i.e., the (i) price willing to pay, (ii) technological complexity and feasibility, (iii) code reuse, (iv) implicit needs of the customer which may not correspond to requirements. Such problems are overcome with a binding agreement.

Table 1. Divergent interests

	Organization	**Contractor**
Requirement interpretation	Broad	Narrow
Time to market	As soon as possible	Depending on several issues
Quality & Security	Best	Good enough to get paid
Cost	As low as possible	As high as possible

However, latent interests are not aligned, due to specification costs, unverifiable actions and dynamic inconsistency. If time and cost are fixed, requirements have a degree of interpretation but they are easily quantifiable, it is quality and security which belongs to a qualitative or "subjective" dimension which is the loosest part of any software contract. Loose quality and security software means unsustainable raising maintenance cost in the long run. Especially mission critical organizations may loose operational capability due to the complexity of their multi-layer system.

Therefore, there is a stringent need in the near future to align organizations and contractors interests, in terms of customer needs, quality & security, costs, and time. The idea is to develop a *bonus-malus* reward system. In such a model, the price is fixed and represents the maximum awardable amount. According to the development process and product quality the contractor is paid according to what is delivered and measured. To do so, there must be a quantifiable measure of some kind of software dimension. With all their limitations, we do believe that function points [2], or simpler variants like simple function points (SFP) [6], represent a fair and quite effective metric. To avoid specification costs, contracts should have a loose - in some way open - requirements list, but a fixed SFP. Moreover, a *bonus-malus* mechanism should be added alongside within the pricing. After each iteration, i.e., implementation of user stories, SFP are consumed and paid. The amount paid follows the *bonus-malus* pricing mechanism. With a high quality code, contractors get a bonus, up to the maximum (fixed) amount.

As any metrics, also SFP has some limitations. For this reason we do not claim that they are the ultimate solution to solve the problem. However, SFP is an easy measurable metric for business functionalities, which are very close to the Agile definition of user story.

Code quality control is still necessary, to avoid the malicious use of low quality functions, just to increase pricing. Therefore, it is of greatest importance to fix such test and metrics within the contract, even if not implemented. Based on our experience, we suggest that security and code quality should be defined as non-functional requirements in the development process. Especially in mission critical organizations we see how some redundancy of competences within the process improves code quality and security [11]. Thus, a TDD (Test Driven Development) approach set in the contract seems quite suitable for Agile contracting. Within each iteration, PO and the contracting development team start with a test oriented development, which has to corresponds to the user story development.

Our main idea is that continuous "tensions" and new equilibria between the two parties are the best mitigation driver, which underlies to any contract. Continuous discussions, bargaining, and agreement do motivate both parties to carry on their respective tasks. In such way we do not have specification costs, since Agile contracts do only specify the very general task and any detail user story development is agreed in any iteration; it is a sort of overarching or framework contract. Dynamic inconsistency is mitigated through a reward based payment. Contractors will have the economic interest to get the "bonus", which is awarded according to their performances. Unverifiable actions are mitigated by a TDD approach, since "quality metrics" i.e., tests, are agreed by the parties within the iterative development process.

Such approach is particularly effective for public administrations which by law must use a bidding base. With such an approach, it is possible to define a budget a priori. At the same time, contractors will work for better quality software, trying to gain the whole amount. Organizations and POs gain from velocity and requirement satisfaction. From an operational point of view, this solution tackles each critical point that Waterfall does not structurally solve.

Finally, from a contractual perspective i.e., the economics of the contract, this solution gets all the benefits of contracting (coordination and exchange implementation). At the same time some major problems of Waterfall contracts (specification cost, dynamic inconsistency and unverifiable actions) are solved by a methodological approach.

3 The Italian Case

Civil law countries face a higher degree of complexity compared to common law ones. The reason is that such countries have to stick to constitutional and written laws, which regulate public procurements. Thus, the structure of the procurement law follows 6usually six constitutionals principles: free competition, equal treatment, non-discrimination, transparency, proportionality, and publicity. These are substantial issues which are always reflected in the procurement law.

According to those principles, the object of the contract, the competition, the economic value, the verification, and the variations are the five pillars on which the procurement law is built.

Although we are now referring specifically to the Italian case, these considerations are of use also for other countries based on civil law. In fact, regulation may slightly change, but the constitutional assumptions and procurement characteristic are basically the same or, at least, comparable. For this reason we believe that this research is of good use also beyond Italian borders.

In the following subsections we will explain how to structure an Agile contract, according to those pillars.

3.1 The object of the contract

The contractual object has to be *determinate* and *determinable*, according to art. 1346 of the Italian civil code (cc). So, the object of the contract needs to be clearly identifiable without further interaction. This means that a collaboration program can not be just agreed upfront, if it is not sufficiently determined. At least, some characteristics of the future software product have to be defined.

According to the procurement law (D.Lgs. nr 50/2016, art. 21.15) the public bid should include a **technical annex**, composed by:

1. Calculation of the alleged cost;
2. Financial statement of total charges;
3. Specific descriptive and performance specifications;
4. Minimum bid requirements;
5. Possible variations;
6. The circumstance of changing the negotiating conditions.

The technical annex is of pivotal importance for Agile contracting, since it is the framework where the (public) customer describes the system and prescribes the methodology. Interestingly, although the procurement law applies easily to Waterfall-like contracts it does not hinder *per se* Agile contracting. However, for these new kinds of contracts the object (the software system which has do be developed) needs to be defined ex ante, with the possibility to refine requirements along the way. This does not appear unreasonable, considering that systems and context requirements are set by regulation or internal policy guidelines. For instance, communication protocols, stakeholders, security standards, interoperability routines and so on are easily known *a priori*.

3.2 The competition

The competition is a key element for public procurements since it guarantees constitutional rights, such as open concurrency, impartiality, and accountability. It is basically a trade-off between such rights and the utility of the contract. In other words, although it would be more effective to deregulate the competition and do it customized and tailor made case by case, constitutional rights needs to be uphold. Thus the competition should ideally be a Pareto-optimal solutions between these opposite forces.

So, it is not rigid *per se*, if it is effectively accountable. The law guarantees certain degrees of flexibility, in order to find the right partner. Therefore, the customer may specify flexible collaboration provisions which meet the Agile philosophy.

3.3 The variations

Variations are of great interest for Agile contracts, since they introduce the necessary flexibility along the contract life. However, although they are possible, they still need to be accountable. Those provisions should be clear precise and unequivocal. Moreover, any variation could be done if one of these cases occur:

1. if the variation causes a modification such that a competitor could had win the competition or more competitors could had participated to the selection process;
2. if the economic equilibrium changes significantly;
3. if the object of the contract is heavily extended.

These are the most important framework boundaries to elaborate an Agile contract.

3.4 The economic value

Also the determination of the economic value has to be clear. Since the economic evaluation is a complex issue, the law admits the idea of flexibility but only with objective and fixed parameters. Once they are set, the price derives from the computation of the produced software and the economic calculation.

Interestingly, the law does not state how to do it. This introduces the right flexibility to develop a proper evaluation techniques for Agile contract. This is also a key issue for the reasons explained in section 2. The most important thing to preserve in an Agile relationship is the alignment of interests. Since most of possible discussion may be around the effective value of the software, identifying an accountable and clear way to define the value, motivates both parties to work together to get the best possible outcome.

3.5 The verification

Finally, also the verification needs to fulfill constitutional requirements. So, the software should be inspected to see if it fulfills the requirements. Such inspection should be accountable and the techniques defined upfront.

This complies very well with the Agile philosophy. Since the verification process is transparent, interests alignment is facilitated. The implementation of non-invasive tools is considered an effective way to enhance accountability along all the development process.

4 Contractualization of Function Points

Function Point Analysis (FPA) [2,20] provides enough objectivity in the evaluation process, independently from the used technology. This is the reason why FPA is a suitable option to guarantee the proper flexibility of the Agile methodology within the Italian constitutional framework discussed before. For the sake of simplification, also novel estimation techniques based on FPA, like simple function points (SFP) [6] may represents a suitable and easy measurable metric, as already discussed in section 2.

Another strong point in favor of Function Points is that these are known and already used within the public sector[4]. This means that it would be rather efficient to build an Agile contract, based on the already acquired experience.

It provides the right *tension* between interests in order to let align them, since it is an accountable process. Moreover, a *bonus-malus* effect would also help towards this direction. Exceptional delivered quality has to be economically recognized, beyond the delivery of functionalities. Similarly, also low quality is discouraged. This mechanism should induce the provider to deliver not just average quality functionalities but high-value ones. Although the functionality is delivered and assessed by FPA, there is no guarantee for quality. In fact, FPA does not assess quality *per se* but if the software compute a certain numbers of functionalities.

For this reason the use of a non-invasive quality tool to assess ongoing quality of software products is of greatest importance. It does not represent a legal issue, since the customer can easily include this methodological requirement in the competition call. Such tool may provide not only the number of developed functionalities but also their quality, according to industrial benchmarks (i.e., ISO/IEC 25010:2011).

So, also the development methodology becomes of importance, since it is complementary to the non-invasive tool. The Test Driven Development (TDD) [4] provides the the right approach to develop mission critical software with highest attention to quality and security.

For this reason we sum up the three keystone of an Agile contract with FPA. In this proposal law and economics aspects of contracts are maximized, upholding constitutional duties of the contracting authority.

1. Specification costs are minimized by the methodology. After several iterations fine-granular functionalities are negotiated.
2. Dynamic inconsistency is is mitigated by a *bonus-malus* mechanism.
3. Non verifiable actions are mitigated by a Test Driven Development and the implementation of non invasive metrics.

These are the main characteristic for a transparent relationship which maximize the contract utility.

5 Contractualization of Scrum Sprints

Another suitable way to write public Agile contracts are sprint-based ones. In this case not Function Points but Scrum sprints are contractualized. So, as in the other case,

[4] `http://www1.interno.gov.it/mininterno/export/sites/default/it/assets/files/22/0011_disciplinare_di_gara.pdf`

functionalities are described at a high level in the object of the contract but the economic value is not determined by the FPA but by the development iterations. It is a sort of body rental contract, where man-hours are organized in sprints. Thus, for a team with 5 people, a sprint of 5 weeks and considering a 40 hours week, each sprint will account for 5000 hour/person. The requirements refinement (through User Stories and continuous iterations) is part of the contract life.

Both parties should be aware of the methodology, not only to avoid misunderstanding but also to prevent miscalculation of the effort. The hope is to build a win-win relationship, where parties are aligned to the goal and are treated fairly. A win-lose solution would be rather suboptimal, since there is no guarantee for a long-term engagement.

1. Sprint definition has to be clear in terms of duration and people. In such contracts people plays the greatest role. The level of expertise, seniority and skills should be taken into consideration while designing Scrum teams.
2. The chosen Agile methodology has to be clear to both customer and provider to organize and setup the development. User Stories estimation is a sensible issue here. An overestimation, as also a underestimation may lead to misinterpretations between the parties as also frustration.
3. The *bonus-malus* mechanism described in the previous section should be clear.
4. The use of monitoring and non-invasive tools is also an important issue for the interests alignment and accountability, as explained in the last section.

6 Conclusions

This paper is a first attempt to carry out a foundational work about Agile contracts. We pointed out how, through the alignment of interests, reduction of asymmetry and flexibility Agile could be wider use in today's software engineering environment, especially within the Public Sector. Moreover we highlighted the keystone for Agile contracting within the Italian public administration. This has a direct impact on all civil law countries, since they face similar procurement law principles.

However, still future work is required and will go in two main directions. Firstly, wider studies about implications and implementation of Agile in legal contracts has to be carry out. Secondly, practical validation of such contracts needs to be studied.

Acknowledgments

The authors would like to thank for partial support: the Italian Ministry of Defense with the PNRM AMINSEP (Agile Methodology Implementation for a New Software Engineering Paradigm definition) project; the Italian Interuniversity Consortium for Informatics (CINI), and CNR-ISTC.

References

1. G. Akerlof. The market for ”lemons”: quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, pages 488–500, 1970.

2. A. Albrecht and J. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering*, 9(6):639–648, 1983.
3. S. Atkinson and G. Benefield. Software Development: Why the Traditional Contract Model Is Not Fit for Purpose. In *Proc. HICSS46, Software Track*, pages 330–339, Hawaii, 2013. IEEE Computer Society Press.
4. K. Beck. *Test Driven Development By Example*. Addison-Wesley, Boston, 2003.
5. P. Ciancarini, A. Messina, F. Poggi, and D. Russo. Agile Knowledge Engineering for Mission Critical Software Requirements. In G. Nalepa and J. Baumeister, editors, *Knowledge Engineering and Software Engineering - Methods, tools, and case studies*, pages 1–21. Springer-Verlag, Berlin, 2017.
6. F. Ferrucci, C. Gravino, and L. Lavazza. Simple function points for effort estimation: a further assessment. In *Proc. 31st ACM Symposium on Applied Computing*, pages 1428–1433, 2016.
7. B. Hermalin, A. Katz, and R. Craswell. The Law and Economics of Contracts. In M. Polinsky and S. Shavell, editors, *Handbook of Law and Economics*, pages 3–138. Elsevier, 2007.
8. M. Lapham, M. Bandor, and E. Wrubel. Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs. Technical Report CMU-SEI-13-TN-31, Software Engineering Institute, Carnegie Mellon University, 2014.
9. M. Lapham et al. Agile Methods: Selected DoD Management and Acquisition Concerns. Technical Report CMU-SEI-11-TN-2, Software Engineering Institute, Carnegie Mellon University, 2011.
10. M. Lapham et al. RFP Patterns and Techniques for Successful Agile Contracting. Technical Report CMU-SEI-13-SR-25, Software Engineering Institute, Carnegie Mellon University, 2016.
11. A. Messina, F. Fiore, M. Ruggiero, P. Ciancarini, and D. Russo. A new Agile Paradigm for Mission Critical Software Development. *Crosstalk - The Journal of Defense Software Engineering*, 29(6):25–30, 2016.
12. R. Myerson. Justice, Institutions, and Multiple Equilibria. *The Chicago Journal of International Law*, 5:91, 2004.
13. K. Nidiffer, S. Miller, and D. Carney. Agile Methods in Air Force Sustainment: Status and Outlook. Technical Report CMU-SEI-14-TN-9, Software Engineering Institute, Carnegie Mellon University, 2014.
14. K. Nidiffer, S. Miller, and D. Carney. Potential Use of Agile Methods in Selected DoD Acquisitions: Requirements Development and Management. Technical Report CMU-SEI-13-TN-6, Software Engineering Institute, Carnegie Mellon University, 2014.
15. A. Opelt, B. Gloger, W. Pfarl, and R. Mittermayr. *Agile Contracts*. Wiley, 2013.
16. S. Palmquist, M. Lapham, S. Garcia-Miller, T. Chick, and I. Ozkaya. Parallel Worlds: Agile and Waterfall Differences and Similarities. Technical Report CMU-SEI-13-TN-21, Software Engineering Institute, Carnegie Mellon University, 2014.
17. R. Posner. Gratuitous Promises in Economics and Law. *Journal of Legal Studies*, 6(2):411–426, 1977.
18. R. Pressman. *Software Engineering: a Practictioner's Approach*. McGraw-Hill, 2014.
19. D. Russo. Benefits of open source software in defense environments. In *Proc. 4th Int. Conf. in Software Engineering for Defence Applications*, volume 422 of *Advances in Intelligent Systems and Computing*, pages 123–131. Springer-Verlag, Berlin, 2016.
20. C. Symons. Function Point Analysis: Difficulties and Improvements. *IEEE Transactions on Software Engineering*, 14(1):2–11, January 1988.
21. E. Wrubel and J. Gross. Contracting for Agile Software Development in the Department of Defense: An Introduction. Technical Report CMU-SEI-15-TN-06, Software Engineering Institute, Carnegie Mellon University, 2015.

22. E. Wrubel, S. Miller, M. Lapham, and T. Chick. Agile Software Teams: How They Engage with Systems Engineering on DoD Acquisition Programs. Technical Report CMU-SEI-14-TN-13, Software Engineering Institute, Carnegie Mellon University, 2014.