Contracting agile developments for mission critical systems in the public sector

Contracting Agile Developments for Mission Critical Systems in the Public Sector

Daniel Russo
University of Bologna, Dept. of Computer Science & Engineering
Bologna, Italy
daniel.russo@unibo.it

Gerolamo Taccogna
University of Genoa, Dept. of Law
Genova, Italy
g.taccogna@unige.it

Paolo Ciancarini
University of Bologna, Dept. of Computer Science & Engineering
Bologna, Italy
paolo.ciancarini@unibo.it

Angelo Messina
Innopolis University
Innopolis, Respublika Tatarstan, Russia
a.messina@innopolis.ru

Giancarlo Succi
Innopolis University
Innopolis, Respublika Tatarstan, Russia
g.succi@innopolis.ru

CCS CONCEPTS

• **Applied computing** → **IT governance**; **Law**; **Economics**; • **Software and its engineering** → **Software creation and management**; Software organization and properties; • **Social and professional topics** → *Professional topics*; *Computing / technology policy*;

KEYWORDS

Software Engineering, Agile, Agile Contracts, Contracting, Law & Economics, Public Sector

ABSTRACT

Although Agile is a well established software development paradigm, major concerns arise when it comes to contracting issues between a software consumer and a software producer. How to contractualize the Agile production of software, especially for security & mission critical organizations, which typically outsource software projects, has been a major concern since the beginning of the "Agile Era." In literature, little has been done, from a foundational point of view regarding the formalization of such contracts. Indeed, when the development is outsourced, the management of the contractual life is non–trivial. This happens because the interests of the two parties are typically not aligned. In these situations, software houses strive for the minimization of the effort, while the customer commonly expects high quality artifacts. This structural asymmetry can hardly be overcome with traditional "Waterfall" contracts. In this work, 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. Finally, a case study is presented, describing how Agile contracting has been concretely implemented in the Italian Defense Acquisition Process. This work is intended to be a framework for Agile contracts for the Italian public sector of critical systems, according to the new contractual law (*Codice degli Appalti*).

1 INTRODUCTION

In several commercial domains Agile development has been effective for building new software systems or to evolve an existing one rapidly, decreasing development costs. The role of IT, intended as value-focused support for the design and implementation of digital technologies, disrupted new product and service developments [31–33]. While consumers are becoming more keen to use technology for their daily applications, businesses are rethinking about customers value and the relative business models for their competitive differentiation [7]. Therefore, entire industries restructured their business processes to deliver new capabilities and goals to the new business model [27]. As software becomes more essential to the world's day-to-day activities, the community is calling to move Agile software development beyond a "craft–based approach to become a true engineering discipline" [16].

In this scenario, a business area of particular relevance for the size and the number of opportunities are mission & security critical systems, especially for defense applications [11]. These applications are usually very expensive and are managed with specific care from public administrations which bid contracts to external software houses. Apparently, a Waterfall software development

process model responds to some fundamental needs in such organizations, like (i) a clear definition of the costs, (ii) early requirement definition, (iii) predefined schedule, and (iv) tracing 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 its limits. Moreover, velocity is a crucial factor for such organizations. Military operations need do be deployed within a very short time range on very different scenarios [12, 23]. Information systems have to evolve accordingly. Waterfall, is not a suitable paradigm, since it is not enough flexible, expensive, and it fails frequently [39].

Although there are industry–scale Agile development techniques for the development of critical systems, like SAFe [13, 21], contractual aspects are typically overlooked. With regard to contractual issues, the adVANTAGE framework is a potential model for commercial organizations [9]. When a project is executed following a contract compliant with the adVANTAGE framework, a priority list of requirements in the form of user stories - namely a backlog - is managed jointly with the customer. Such a backlog is addressed in a sequence of Sprints. Each Sprint is a project phase of fixed duration, usually from two to four weeks, and realizes the software satisfying some of the top requirements found in the backlog. The development services are billed according to the quality of the software delivered at the end of each Sprint. If the contractor fails to realize the user stories agreed for a certain Sprint within the available jointly agreed target budget, the additional costs will be charged at reduced daily rates. This approach or one similar is often followed in contracts between private companies.

However, the lack of concrete proposals, their experimentations, and discussions in technical and scientific forums of Agile contractual models suitable also for the public sector is one of the reasons why top managers are not keen to this development method. Accordingly, this work is an attempt to overcome such limitation.

We propose a foundational approach connecting the theory of how contracts should be organized with Agile practices, using the Italian context as a reference, and also identifying the key issues of Agile contracting which need further development.

The paper has the following structure. Related work are presented in Section 2. In Section 3 Law & Economics of contract theory are 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 4 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 approaches (Section 5); the first one (5.1) is based on Function Point Analysis; the second (5.2) is based on Scrum Sprints. A case study detailing how Agile contracting has been concretely implemented in the Italian Defence Acquisition Process is described in Section 6. Finally, in Section 7 we sum up our main proposal and envision some further work.

2 RELATED WORK

The problem of Agile contracting is old. Basically, the main difference with respect to contracts for Waterfall developments, that are based on measuring and compensating effort spent during the process, is that fixed price seems more adequate for agile developments. The Agile fixed price is a contractual model which includes an initial phase after which budget, delivery dates, and the way of defining the scope of the system being built is agreed upon.

For instance in 2006 Alistair Cockburn published online an intriguing discussion of some typical Agile contracts[1]. His page lists several possibilities, like the following ones:

- fixed price, fixed scope, fixed time;
- fixed price, fixed time, negotiable scope;
- paying for effort as it gets spent. If the requirements are volatile and there is mutual trust among producer and consumer this is the best situation;
- max price with payment on incremental acceptance: it works with stable requirements;
- incremental delivery with payment on incremental acceptance
- price for each unit delivered, for instance a fixed fee for function point;
- base fee for each unit delivered, plus a low fee per hour, in order to incentive developers to early delivery.

Similar contractual cases are also discussed in the work of Pilios [30]. Further aspects of an Agile contracts are risk share (customers and developers compensate the additional expenses for unexpected changes equally amongst themselves) or the option of either party leaving the contract at any stage (exit points).

Indeed, Agile methods tackle those issues, trying to align the interests of the development team and the customer. Our interest here is for Agile applied to mission critical systems sponsored by public institutions, with specific application to Italian public institutions.

In some earlier works we have reviewed the enactment of Agile software development methods within mission & security critical organizations, especially military ones [12, 23, 36, 37]. Moreover, in the last years the debate around the use of Agile contracting also for commercial uses became a trending topic [5, 8, 9, 28]. In [5] the authors criticize traditional contracts for software development, which increase the risk of failure because requirements are frozen due to the sequential work flow, leading to a low quality of design, and causing a poor return on investment. The book [28] suggests fixed price or maximum price contracts for Agile developments, in contracts structured as follows. The contract has to describe to what extent, in percentage, the costs incurred by the supplier will be charged to the customer when the maximum price range is exceeded. A period of n Sprints is agreed upon as the test phase of cooperation. The final milestone is a checkpoint whereby the customer and supplier can enter into the real development of the project or maybe exit in a controlled manner. Another, more recent book [9] expanded a contractual model called adVANTAGE for Agile Developments, already sketched in [8]. This model puts specific focus on the willingness of the contractor to take some of

[1] http://alistair.cockburn.us/Agile+contracts, retrieved on September 19, 2017

the (apparent) risks of development that come with Agile practices. Still, it is not suited for public administrations, since it does not consider constitutional specificities, outlined in Section 4.

Recently the US government has devoted a lot of attention to the problem of Agile contracts. The Software Engineering Institute (SEI) has released in the last five years several reports concerning Agile for producing software products in particular for the military [17, 19, 25, 26, 29, 42]. It has also published some guidelines for Agile contracts for software acquired by the US DoD [18, 41]. 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 SQALE [22] for measuring technical debt in terms of effort, or bug defect rates, length of throughput time compared to contractor estimates, speed of time to value.

Nevertheless, despite these efforts, the problem of understanding how to lay down contracts for Agile development for mission critical products is still at its infancy.

3 THE LAW & ECONOMICS OF AGILE CONTRACTS

Contracts are agreements between two parties, with different interests, written down to fix such interests, alongside with some 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 [15]. 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 party's 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 [34].

From a Law & Economics viewpoint, there are several issues regarding the importance of contracting [15].

- *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. Without coordination, 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 reduce the imprisonment time as a benefit). With coordination, on the contrary, both would get the better payoff, not admitting the crime, gambling the law system, escaping from a long imprisonment time. If 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 excellent models to understand institutions [24].
- *Exchange implementation.* Especially in situations of hidden informations (i.e., information asymmetry occurring when one party has an information which the other party does not have), contracts may mitigate such asymmetry [2]. To avoid adverse selection, 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 licensed 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 [15]. The most important from our point of view are:

- *Ex post: specification cost.* Writing down 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, since contingencies are not defined precisely enough. Anyway, in both scenarios, contracts fail to assure the commitment of the parties.
- *Ex ante: dynamic inconsistency.* This is the classic investment problem. One party may be willing to bargain and to modify the contracts when it has pursued investments. Suppose that a vendor has found that the software can easily have more functionality or higher quality even with limited additional costs. However, the price has been set, so the motivations to deliver such higher value are minimal if the customer is not ready to agree, which is in turn difficult because the intangible nature of software may make difficult for the customer to understand the nature of such modifications (see the next point). In essence, vendors may not have any incentives to do investments, i.e., spend money to develop high quality code.
- *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 assess with objectivity if what promised has been carried out according to the contract.

While we analyse Agile contracting, we should avoid the risk of overstating its problems and overlooking 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 the Law & Economics theories of contracts is that any contract has its loopholes, thus also Waterfall ones.

Waterfall contracts are well known, in a sense they are easy to agree initially, because parties are usually fully aware of them and there is a history of people using them - meaning that it is more difficult for a manager to be subject to criticism for adopting them[2]. Both specification costs and unverifiable actions have a big impact on the 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 to build. Such barriers have a direct impact on both the contract cost and market efficiency. In Waterfall contracts there are indeed "hidden" costs that indirectly increase the cost of software products. The perceived "reliability" of Waterfall has apparently scarce evidence in practice. What we do know is that Waterfall usually increases the maintenance costs, which are hidden costs belonging to the software's life cycle [35]. However, as mentioned, while there are established routines concerning how to carry out a Waterfall contract, there are very few guidelines about Agile.

First of all we will depict the divergent interests of a software contract, represented in Table 1. As seen before, contracts facilitate market equilibrium through coordination and exchange implementation. In software this means that the two parties which suffer from an information asymmetry reach an agreement through a legal binding paper (the contract). A generic organization does not always have the expertise or the man-power to carry out the software, while contractors do. There is asymmetry in the sense that both parties are not aware of the same relevant information, 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, some latent interests are not aligned by any contract, 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 belong to an arbitrary or "subjective" dimension which are the most difficult parts to fix in 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 and low quality of their multi-party systems. Therefore, there is a stringent need in any field to align organizations and contractors interests, in terms of customer needs, quality & security, costs, and time.

From a project management perspective, divergent interests can also be explained with the Iron Triangle [4], represented in Figure 1. Scope, schedule and budget are typically opposite concepts, where the optimum is represented by a balanced relation [4]. The variable that does not change is quality, which is the major concern of every software project. A software project with a broad scope, and tight schedule and budget will reasonably be of poor quality. Similarly, in the case of high budget availability but with a stiff schedule and broad scope will also deliver poor quality software due to general low predictability of software project management for short time frames. Finally, the scenario where a software project has to be completed within a short time and low budget with a broad scope is quite unrealistic. Still, the Iron Triangle is extremely useful to understand the three project management drivers for a software system with certain quality standards. When parties with divergent interests engage in a contractual relationship, both need realistic expectations. An unbalanced relationship, which may be considered with favor in a first moment, will unlikely deliver a satisfactory outcome, in terms of software quality. Consequently, the substantial aim of Agile contracting is both to understand and to align such divergent interests. However, those interests are rarely clear ex ante, at the beginning of the contractual relation. Therefore, the continuous specification of those interests are pivotal for a successful project outcome. Clearly, this wisdom fits particularly well to systems which scope evolve in time. So, to tailor the scope (i.e., requirements specification), along with adequate budget and schedule, multiple contractual agreements are needed to formalize each Agile iteration. Consequently, we propose in Section 4 a *double–level contracting structure* to enhance the ongoing interest alignment.

Our 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 obtained, 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 size dimension. With all their limitations, we do believe that Function Points [3, 38], or some related variants like *Simple Function Points* (SiFP) [14], represent a measure that is good enough for our purposes. To avoid specification costs, contracts should have a loose - in some way open - requirements list, but a fixed, predetermined SiFP estimate. Moreover, a *bonus-malus* mechanism should be added alongside within the pricing. After each iteration i.e., implementation of user stories, SiFP are consumed and paid. The *bonus-malus* pricing mechanism in this case means that with a high quality code, contractors get a bonus, up to the maximum (fixed) amount. Instead if they deliver a release which exposes some technical debt contractors get some kind of fine, to

[2]Remember the old adage "None has been fired for using IBM." which fully expresses the criticism manages can have by trying new, not yet consolidated, approaches. This phenomena is also exploited by vendors as it is well explained in the *Wikipedia* page on "Fear, uncertainty and doubt" accessed at URL: https://en.wikipedia.org/wiki/Fear,_uncertainty_and_doubt on Sept. 10, 2017.

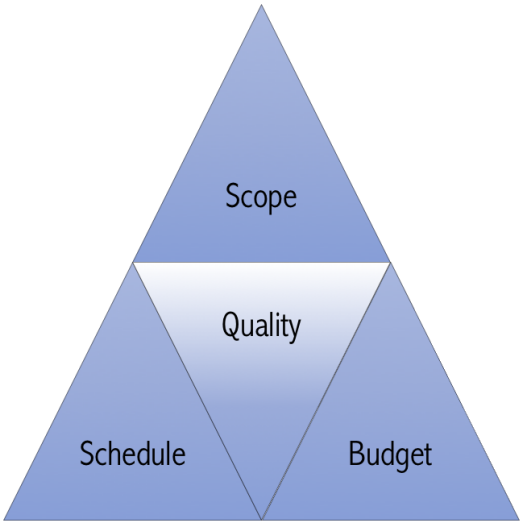

Figure 1: Iron Triangle [4]

be recovered after the debt is repaid. Clearly, a critical issue is how to measure the technical debt. Some modern tools like SonarQube are able to measure technical debt [10]. As any metrics, both FP and SiFP have some limitations. For this reason we do not claim that they are the ultimate solution to solve the problem. However, SiFP 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 [23]. Thus, a TDD (Test Driven Development) approach set in the contract seems quite suitable for Agile contracting. Within each iteration, the Product Owner (PO) and the contracting development team start with a test oriented development, which has to correspond to the user story development.

Our main idea is that continuous "tensions" and new equilibria between the two parties are the best mitigation drivers that underly to any contract. Continuous discussions, bargaining, and agreement do motivate both parties to carry on their respective tasks. In such way, we think that we can obtain the following results:

- Specification costs are limited, since Agile contracts do not only specify the very general task at the beginning but they agree the details of every user story upfront each iteration; this is a sort of overarching or framework contract.
- Dynamic inconsistency is attenuated 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 our law must use a bidding base. Following our proposal it is possible to define a budget a priori and, at the same time, contractors will work for better quality software, trying to gain the whole amount. Organizations and project owners 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, namely coordination and exchange implementation. At the same time some major problems of Waterfall contracts (specification cost, dynamic inconsistency and unverifiable actions) are substantially reduced.

4 THE ITALIAN CASE

Although we are now referring specifically to the Italian case, these considerations are of use also for other countries based on European public procurement rules. 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.

The structure of the procurement law follows some basic constitutional principles, comprehending:

(1) free competition,
(2) equal treatment and non-discrimination,
(3) transparency,
(4) adequacy and proportionality.

These are substantial issues, which are always reflected in any concrete application of the law. In the following subsections we will try to explain how to structure an Agile contract, according to those pillars.

4.1 The object of the contract

The contractual object has to be *determinate* or *determinable*, according to art. 1346 of the Italian civil code (cc in short). So, the object of the contract needs to be clearly identifiable without further arbitrary decisions. 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. Moreover, according to the procurement law (D.Lgs. nr 50/2016, art. 23.15) 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) awarding criteria;
(6) possible variations;
(7) the possible circumstances of (non substantial) change of the negotiating conditions.

The technical annex is of pivotal importance for Agile contracting, since it is the document (or the set of documents) where the

public customer describes the required system and prescribes the methodology. Interestingly, the procurement law applies easily to Waterfall-like contracts but does not hinder Agile contracting *per se*. Consequently, the object of Agile contracts (i.e., the software system to developed, evolve or maintain) needs to be defined *ex ante* at least in functional terms, with the possibility to refine requirements along the way. Thus, a corresponding well fitting structure of the contractual relationship is now proposed.

4.2 The structure of the contract

Our proposed solution for Agile contracts consists of a **double-level contracting**. To clarify our idea, see Figure 2, which exemplifies the proposed structure.

In European public procurement law, this is allowed by the rules concerning the framework agreements. According to article 33 of the directive 2014/24 EU, a contracting authority may conclude such an agreement, observing the procedures provided by the directive (i.e. the ordinary awarding procedures). In fact:

> "*In general terms a framework agreement means an agreement between one or more contracting authorities and one or more economic operators, the purpose of which is to establish the terms governing contracts to be awarded during a given period, in particular with regard to price and, where appropriate, the quantity envisaged.*" §33.1, Directive 2014/24 EU.

The purpose is to establish the two–level contractual governance. In our case, such a structure between one or more authorities and a single operator is appropriate. The first level contract defines, in general terms, all customer's needs, and in particular the context in which the software will be used to meet such needs:

- the high level definition of software's functionalities and architecture;
- the quality and performance standards to be reached (minimum standards and higher ones, possibly to be paid more by the customer);
- the criteria for the the product evaluation and possibly the definition of done;
- the time frame;
- the general terms of the compensation.

The framework agreement should be limited in time: e.g., not exceeding 4 years. With second level contracts, parties agree the specifications within a variable number of iterations, considering the points listed before, as sketched in Figure 2. Before each iteration both parties fine tune the first level issues, in order to meet iteration's scope. In particular, the object of each iteration is specified, along with software's functionalities, performance standard, product evaluation, terms of compensation and time frame. In other words, the framework agreement (first level contract) opens the way to a number of subsequent detailed contracts for the execution of the whole project, each of which can be adjusted according to the results of the previous ones.

The mentioned public procurement rules permit the awarding of such single contracts to the contractor, under the framework agreement, without any new competitive procedure. The only exception is if the contracting authority changes, or if the object of the framework agreement is substantially modified. The key provision therefore can be found in the European directive, according to which:

> "*Where a framework agreement is concluded with a single economic operator, contracts based on that agreement shall be awarded within the limits of the terms laid down in the framework agreement*" and "*For the award of those contracts, contracting authorities may consult the economic operator party to the framework agreement in writing, requesting it to supplement its tender as necessary*" §33.3, Directive 2014/24 EU.

Indeed, the possibility to design a multi–level contract, where specifications are negotiated before each iteration, is an important legal tool for Agile purposes. From our point of view, supplementing the tender would mean negotiating and finding an agreement on the fine tuning aspects necessary before each one of the progressive functional steps or scrum sprints Moreover, such a double-level structure of the contractual relationship between the customer and the contractor has the advantage that the framework agreement does not oblige the customer to engage in the second level contracts.By this way, the customer always has the power to terminate the relationship, after each iteration. This may happen for several reasons, like for instance any dissatisfaction of the relationship, although the general budget of the framework agreement has not been fully spent.

4.3 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 through *ad hoc* designs, constitutional rights need to be uphold. Thus, the competition should ideally be a Pareto-optimal solution between these contrasting forces. The law guarantees certain degrees of flexibility, in order to find the best partner.

In our case, such an opportunity arises from the fact that the competitive awarding procedure would be limited at the level of the framework agreement with its above indicated general elements of the project. In practice, an efficient and effective selection of the general elements is pivotal to make sure that the advantages of the best offer will be fully reflected afterwards. Doing so will assure that the second level contracts will be best suited for the general scope of the software system, considering the afterwards non competitive negotiation between the parties for the fine tuning of any iteration. Achieving such a goal needs some caution, especially about the link between quality and performance standards, on one hand, and compensation on the other.

4.4 Economic value

The determination of the economic value has to be clear and effective. In the case of framework agreements and subsequent second level contracts, it is possible and appropriate to set binding general rules about compensation. This should be clearly stated in the framework agreement, and applied in the negotiation of the second level contracts. Thus, in the first level contract, the parties should

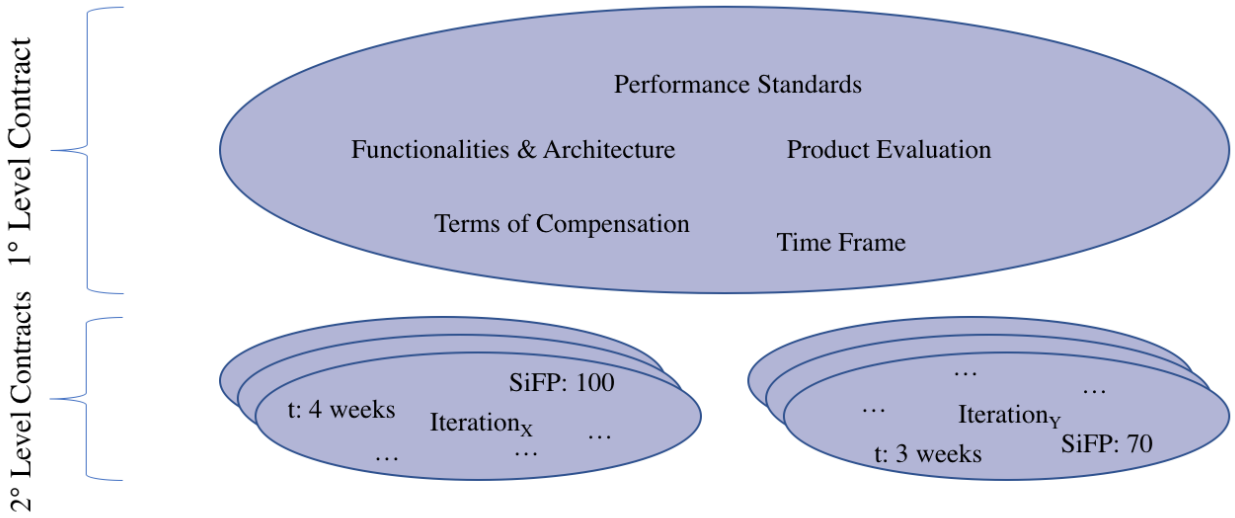

Figure 2: Structure of the contract

fix the compensation structure, detailing the cases of special awards and penalties.

As an example, they could agree that for each iteration, a mix of a fix rate and a decremental person-day rate (in order to encourage the contractor to be efficient in execution) could be negotiated. Moreover, a quality-related awarding system may also be helpful to enhance project's quality, such as a *bonus-malus* mechanism. On such a basis, second level contracts that are continuously negotiated before each further iteration should respect the general awarding schema. However, small modification to adjust specific necessities may be further negotiated to fully align divergent interests. In other terms, once the general criteria are set in the first level contract, the price for each iteration derives from the corresponding negotiated estimation of the cost of the software system, along with the upon agreed calculation criteria. This means that proper evaluation techniques for Agile contracts are not only possible, but also recommended for the reasons explained in Section 3. Indeed, the most important issue to preserve in an Agile relationship is the alignment of interests. Since most of possible discussions may be around the effective value of the software, identifying an accountable and clear way to define the economic value, within the exposed provisions concerning compensation of the contractor, motivates both parties to work together to get the best possible outcome.

4.5 Provision of accountable variations

Variations are of great interest for Agile contracts, since they introduce the necessary flexibility along contract life. Generally speaking, public procurement rules make variations of contracts possible, but with clear limits. Any variation should be forbidden if one of these cases occur:

(1) if the variation causes a modification such that a competitor could had won the competition, or if other competitors could had participated to the selection process;
(2) if the economic equilibrium of the contract changes significantly;
(3) if the object of the contract is heavily extended and/or modified.

This applies to the framework agreement and to any amendment of it. Fine tuning of second level contracts would not mean any variation. Our solution simply consists in limited specifications concerning each iteration. Of course, variations may be possible in the case that the system's scope of the iteration is consistently different with the first level contract and can not match the general terms. For such an event, the general limits to variations should be respected. Although this legal tool is an element of flexibility, a parsimonious use of it is recommended. In fact, a frequent use of variation may harm the general contractual governance. This would introduce opacity in the relationship which should avoided to enhance information symmetry between the parties.

4.6 Verification

Finally, also the verification needs to fulfill some legal requirements. Once built, or even during its development, the software should be inspected to see if it fulfills the software's scope. 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.

5 SETTING UP THE CONTRACTS

In this section of the paper we are going to sketch two possible implementation of our proposal, based on the contracting of Function Points and Scrum Sprints. These two examples should be embedded in the first level contract to determine the terms of compensation.

A key point for the contractual governance is the agility of the assessment of the economic value. Too complex metrics are not suited for such an environment, since it clashes with the *need for speed* of Agile iterations. Ideally, any assessment metric which assure fast and reliable (i.e., objective) outcomes is suitable for this purposes. Accordingly, we propose here two valuable but different examples. Functional size measurement methods focus on the code, while Scrum Sprints on the process. Clearly, both have upsides and downsizes briefly discussed. Although there may not be one best solution, rather several Pareto–optimal ones, it is of pivotal importance to understand the own contractual environment under which the software system is developed to use one of the proposed solutions or even new ones.

5.1 Contracts with Function Points

Function Point Analysis (FPA) [3, 40] 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 (SiFP) [14, 20], may represent a suitable and easy measurable metric, as already discussed in Section 3.

The definition process of SiFP - a quite straightforward functional size measurement method - is represented in Figure 3. The idea behind SiFP is to provide a good evaluation of the functional size of an application, without redundant basic functional components (BFC) i.e., DET - Data Element Type, FTR - File Type Referenced, and RET - Record Element Type. Therefore, only two BFCs are needed, according to this method:

- Unspecified Generic Data Group (UGDG)
- Unspecified Generic Elementary Process (UGEP)

Consequently, BFCs are of only two types: data object type, and transactional object type. Hence, the size of any software application may be expressed as the number of SiFP:

$$SiFP = M_{UGDG} + M_{UGEP} \tag{1}$$

Where M_{UGDG} and M_{UGEP} indicate the measures of the size of BFCs, as a result of IFPUG measurement rules. Interestingly, FPA measurements take into consideration e.g., new development, functional enhancement, and software assets, which SiFP does not. Still, SiFP are convertible to traditional FPA (like IFPUG), due to the high correlation through a regression analysis [1]. Therefore, from a practical perspective SiFP are convertible to any FPA, since functional size measurement methods are strongly correlated from a structural point of view.

Moreover, besides the simplification of BFCs, measurement weights are fixed, with respect to IFPUG. In particular, [20] propose that:

$$SiFP = 4.6UGDG + 7UGEP \tag{2}$$

Where SiFP are 4.6 times the number of elementary processes, without considering the primary intent, and 7 times the number of logical data files, without considering if they are internal or external.

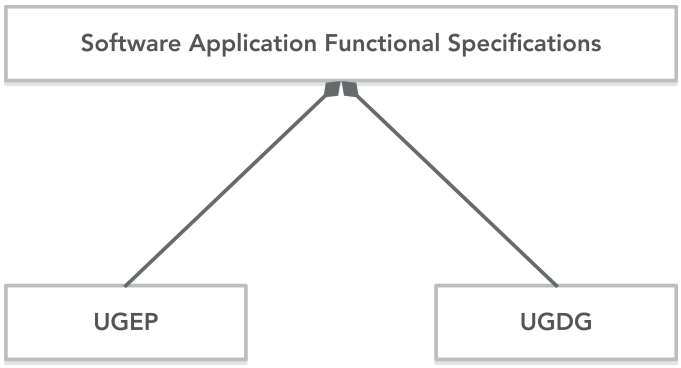

Figure 3: SiFP definition structure

Another strong point in favor of Function Points is that these are known and already used within the Italian public sector[3]. This means that it would be quite effective to write an Agile contract, based on the already acquired experience. FPA 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. This mechanism should induce the provider to deliver not just average quality functionalities but high-value ones. We remark that although the delivered functionality can be first estimated and then assessed by FPA, there is limited guarantee for quality. In fact, FPA does not assess quality itself but only if the software computes a certain number of functionalities. Exceptional delivered quality has to be economically recognized, beyond the delivered functionalities. Similarly, also low quality should be discouraged. 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 a tool may compute not only the number of developed functionalities but also judge their quality, according to industrial benchmarks (i.e., ISO/IEC 25010:2011). An example of such a tool is SonarQube [10].

So, also the development methodology becomes of importance, since it is complementary to the non-invasive tool. Furthermore, the Test Driven Development (TDD) method [6] provides a useful approach to develop mission critical software with the highest attention to quality and security. For this reason we now sum up the three keystones of an Agile contract with FPA. In our proposal Law & 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.

[3] See for instance http://www1.interno.gov.it/mininterno/export/sites/default/it/assets/files/22/0011_disciplinare_di_gara.pdf retrieved on Aug. 23, 2017

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

5.2 Contracts with Scrum Sprints

Another suitable way to write Agile contracts for the public administration are Sprint-based ones. In this case, Scrum Sprints are the base for terms of compensation. So, as in the other case, 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 for an iteration of 5 weeks (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 aim 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 temporal duration and people committed. In such contracts people play the greatest role. The level of expertise, seniority, and skill should be taken into consideration while designing Scrum teams.
(2) The chosen Agile method 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, but 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 clearly stated.
(4) The use of monitoring and non-invasive tools is also an important issue to both interests alignment of all parties and improving accountability, as explained in the last section.

6 CASE STUDY

The ideas exposed in this paper have been elaborated and evaluated in a contract with the Italian Defense Acquisition Process, partially described in [23]. For this project we did not use the proposed contractual schema, although some principles were implemented (e.g., the *bonus-malus* mechanism).

The reference project has been the LC2Evo technology demonstrator. The project lasted about 87 weeks of work, that is, about two years. Sprints lasted 4 or 5 weeks each. Overall, the cost of the project was about 2.6 Million Euro. In this case instead of Function Points, Scrum Sprints have been used to define the contract. After each Sprint, the Definition of Done and acceptance criteria were assessed by external Army engineers. Their assessment were also useful for the subsequent Sprint planning.

A generic +50% increment factor was applied to take into account the better effectiveness and the reduced risk associated to the LC2Evo improved Agile development environment, with respect to the previous Waterfall development cycle.At the end of the project this +50% increment factor has proven to be pessimistic, since the effectiveness of the improved Agile development cycle was much

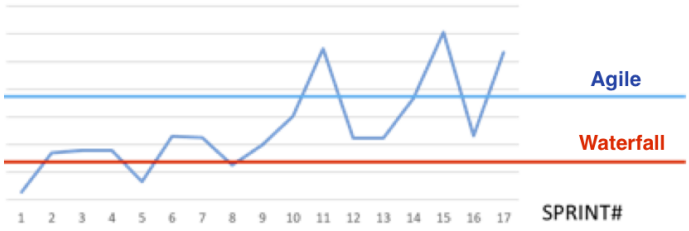

Figure 4: Effectiveness of the contract structure

better than expected. Moreover, due to an important cost reduction, leftover funds where used for extra hardware improvements.

Using a time and material estimation for a predetermined number of iterations (six to eight as we refer to LC2Evo) may be seen as a sort of "body rental" contract, where man-hours and materials are allocated in Sprints. The requirements refinement (through User Stories evolution and continuous iterations) was included in the contract. Both parties were trained and fully aware of the Scrum method, to avoid misunderstanding and prevent miscalculation of the effort. Figure 4 summaries the results that have been obtained using this approach. Unfortunately, specific details are classified for national security reasons. However, the resulting proportion provides a fairly accurate visualization of what happened. After an initial phase, the productivity increase of the new Agile approach was quite evident. Still, the merit of such productivity improvement can not be merely attributed to the different contracting structure, since the project followed a more traditional Arms & Materials contract. However, organizing the contract in an Agile fashion was a clear prerequisite for the project's success.

Finally, this case study gave us the relevant experience to design the new Agile contractual scheme, suitable for the Italian public administrations.

7 CONCLUSIONS

This paper is an attempt to carry out a foundational work about Agile contracts. Starting from the Law & Economics of contracts, we explained how relevant principles should be wider understand by the software engineering community. 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. Indeed, the awareness of software engineers about the economic motivations behind a contractual relationship is of pivotal importance to enhance software quality. Information symmetry is of greatest importance for Agile software development, due to structural reasons related to the day-by-day relationship with the customer and the short development–deployment iterations.

Companies which strive for competitive advantages can easily customize Agile contracts according to the contractual freedom principle (i.e., art. 1322 cc). This is not the case for public administrations which need to follow strict constitutional duties regarding public procurements. However, also these organizations (especially in the defence & security domain) need their software systems to evolve rapidly, to address new-world scenarios. Therefore, this

paper provides the first discussion about Agile contracting for the Public Sector.

Doing so, we highlighted the keystone for Agile contracting within the Italian public administration. These recommendations have a direct impact on all civil law countries, since they face similar procurement law principles. To contextualize better our proposal, we used a case study where Agile contracting principles has been implemented. The outcomes of the presented project are positive and significant.

Future work will proceed in two main directions. Firstly, both foundational as empirical studies about the implications and implementation of Agile contracts will be carried out. Accordingly, a detailed framework for an Agile contract for the Italian public administration will be presented in the near future. Secondly, the empirical validation of such contracts needs to be further studied. In particular elements related to the assessment of non-invasive measurements, effort estimation based on Sprints or SiFP, as well as social aspects of the negotiation should be considered for further studies.

ATTRIBUTION

This paper is a joint contribution of all authors which are listed in order of contribution. Authors contributed according to their different domain knowledge in an interdisciplinary fashion.

ACKNOWLEDGMENTS

The authors would like to thank Col. Franco Cotugno – SEGRED-IFESA/DNA, for the initial idea at the basis of this paper. We also thank for their substantial support: PRIN GAUSS (Governing Adaptive and Unplanned Systems of Systems); 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), the Institute of Cognitive Sciences and Technologies of the Italian National Research Council, and Innopolis University.

REFERENCES

[1] A.Z. Abualkishik, F. Ferrucci, C. Gravino, L. Lavazza, G. Liu, R. Meli, and G. Robiolo. 2017. A study on the statistical convertibility of IFPUG Function Point, COSMIC Function Point and Simple Function Point. *Information and Software Technology* 86 (2017), 1–19.

[2] G. Akerlof. 1970. The market for "lemons": quality uncertainty and the market mechanism. *The Quarterly Journal of Economics* (1970), 488–500.

[3] A. Albrecht and J. Gaffney. 1983. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering* 9, 6 (1983), 639–648.

[4] R. Atkinson. 1999. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management* 17, 6 (1999), 337–342.

[5] S. Atkinson and G. Benefield. 2013. Software Development: Why the Traditional Contract Model Is Not Fit for Purpose. In *Proc. HICSS46, Software Track*. IEEE Computer Society Press, Hawaii, 330–339.

[6] K. Beck. 2003. *Test Driven Development By Example.* Addison-Wesley, Boston.

[7] S.J. Berman. 2012. Digital transformation: opportunities to create new business models. *Strategy & Leadership* 40, 2 (2012), 16–24.

[8] M. Book, V. Gruhn, and R. Striemer. 2012. adVANTAGE: A fair pricing model for agile software development contracting. In *Agile Processes in Software Engineering and Extreme Programming*, C. Wohlin (Ed.). Springer, Malmo, Sweden, 193–200.

[9] M. Book, V. Gruhn, and R. Striemer. 2016. *Tamed Agility.* Springer.

[10] G. Campbell and P.Papapetrou. 2013. *SonarQube in Action.* Manning Publications.

[11] P. Ciancarini, S. Litvinov, A. Messina, A. Sillitti, and G. Succi (Eds.). 2018. *Proc. 5th Int. Conf. on Software Engineering for Defense Applications.* Advances in Intelligent Systems and Computing, Vol. 717. Springer.

[12] P. Ciancarini, A. Messina, F. Poggi, and D. Russo. 2017. Agile Knowledge Engineering for Mission Critical Software Requirements. In *Knowledge Engineering and Software Engineering - Methods, tools, and case studies*, G. Nalepa and J. Baumeister (Eds.). Springer-Verlag, Berlin, 1–21.

[13] C. Ebert and M. Paasivaara. 2017. Scaling Agile. *IEEE Software* 6 (2017), 98–103.

[14] F. Ferrucci, C. Gravino, and L. Lavazza. 2016. Simple function points for effort estimation: a further assessment. In *Proc. 31st ACM Symposium on Applied Computing*. 1428–1433.

[15] B. Hermalin, A. Katz, and R. Craswell. 2007. The Law and Economics of Contracts. In *Handbook of Law and Economics*, M. Polinsky and S. Shavell (Eds.). Elsevier, 3–138.

[16] I. Jacobson, I. Spence, and E. Seidewitz. 2016. Industrial-scale agile: from craft to engineering. *Commun. ACM* 59, 12 (2016), 63–71.

[17] M. Lapham et al. 2011. *Agile Methods: Selected DoD Management and Acquisition Concerns.* Technical Report CMU-SEI-11-TN-2. Software Engineering Institute, Carnegie Mellon University.

[18] M. Lapham et al. 2016. *RFP Patterns and Techniques for Successful Agile Contracting.* Technical Report CMU-SEI-13-SR-25. Software Engineering Institute, Carnegie Mellon University.

[19] M. Lapham, M. Bandor, and E. Wrubel. 2014. *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.

[20] L. Lavazza and R. Meli. 2014. An evaluation of simple function point as a replacement of IFPUG function point. In *Proc. Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement.* IEEE, 196–206.

[21] D. Leffingwell. 2016. *SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering.* Addison-Wesley Professional.

[22] J.L. Letouzey and M. Ilkiewicz. 2012. Managing technical debt with the SQALE method. *IEEE Software* 29, 6 (2012), 44–51.

[23] A. Messina, F. Fiore, M. Ruggiero, P. Ciancarini, and D. Russo. 2016. A new Agile Paradigm for Mission Critical Software Development. *Crosstalk - The Journal of Defense Software Engineering* 29, 6 (2016), 25–30.

[24] R. Myerson. 2004. Justice, Institutions, and Multiple Equilibria. *The Chicago Journal of International Law* 5 (2004), 91.

[25] K. Nidiffer, S. Miller, and D. Carney. 2014. *Agile Methods in Air Force Sustainment: Status and Outlook.* Technical Report CMU-SEI-14-TN-9. Software Engineering Institute, Carnegie Mellon University.

[26] K. Nidiffer, S. Miller, and D. Carney. 2014. *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.

[27] OECD. 2016. Stimulating digital innovation for growth and inclusiveness. (2016).

[28] A. Opelt, B. Gloger, W. Pfarl, and R. Mittermayr. 2013. *Agile Contracts.* Wiley.

[29] S. Palmquist, M. Lapham, S. Garcia-Miller, T. Chick, and I. Ozkaya. 2014. *Parallel Worlds: Agile and Waterfall Differences and Similarities.* Technical Report CMU-SEI-13-TN-21. Software Engineering Institute, Carnegie Mellon University.

[30] E. Pilios. 2015. *Contracting practices in traditional and agile software development.* Ph.D. Dissertation. University of Leiden, NL.

[31] M. Porter and J. E Heppelmann. 2014. How smart, connected products are transforming competition. *Harvard Business Review* 92, 11 (2014), 64–88.

[32] M. Porter and J. E Heppelmann. 2015. How smart, connected products are transforming companies. *Harvard Business Review* 93, 10 (2015), 96–114.

[33] M. Porter and V. E. Millar. 1985. How information gives you competitive advantage. *Harvard Business Review* 63, 4 (1985), 149–160.

[34] R. Posner. 1977. Gratuitous Promises in Economics and Law. *Journal of Legal Studies* 6, 2 (1977), 411–426.

[35] R. Pressman. 2014. *Software Engineering: a Practictioner's Approach.* McGraw-Hill.

[36] D. Russo. 2016. Benefits of Open Source Software in Defense Environments. In *Proc. 4th Int. Conf. in Software Engineering for Defence Applications (Advances in Intelligent Systems and Computing)*, Vol. 422. Springer-Verlag, Berlin, 123–131.

[37] D. Russo, V. Lomonaco, and P. Ciancarini. 2018. A Machine Learning Approach for Continuous Development. In *Proceedings of 5th International Conference in Software Engineering for Defence Applications.* Springer, 109–119.

[38] C. Santana, F. Leoneo, A. Vasconcelos, and C. Gusmao. 2011. Using Function Points in Agile Projects. In *Agile Processes in Software Engineering and Extreme Programming (Lecture Notes in Business Information Processing)*, Vol. 77. Springer, 176–191.

[39] StandishGroup. 2016. The CHAOS report. (2016). http://www.standishgroup.com/outline

[40] C. Symons. 1988. Function Point Analysis: Difficulties and Improvements. *IEEE Transactions on Software Engineering* 14, 1 (January 1988), 2–11.

[41] E. Wrubel and J. Gross. 2015. *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.

[42] E. Wrubel, S. Miller, M. Lapham, and T. Chick. 2014. *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.