NTNU
Kunnskap for en bedre verden

Contractual Incompleteness - bug or feature?

An investigation of deliberately incomplete contracts in software development projects

Marit Rossnes

MTM
Oppgaven levert: januar 2014
Hovedveileder: Arild Aspelund

Norges teknisk-naturvitenskapelige universitet
Institutt for industriell økonomi og teknologiledelse

Abstract

This thesis investigates contractual incompleteness in software development projects. Specifications requirements and development effort estimates are traditionally at the core of software development contracts and it is this kind of contractual incompleteness I have investigated in a multiple case study of four Norwegian software development projects.

Removing specifications and estimates is a radical proposition; it challenges established notions of accountability, of investment and of control. It's as such not a viable or feasible option in every context. Academic research on Contractual Incompleteness and my empirical research points towards it being a possible and valuable practice in cases where there is trust between the contracting parties, where there is considerable uncertainty, volatility and ambiguity. Theories on cognitive biases in addition tell us that our ability to predict and plan is limited.

Deliberate contractual incompleteness seems to be valuable to the projects in my study in the sense that they are able to grow their product, their software more organically, build and learn continuously. An early and small increment is used in order to understand what the real requirements are. But in the process of shedding some traditional planning practices, they also seem at risk of losing sight of the big picture. Practices for these kinds of projects then need to deal with both seeing the big picture and facilitating a more organic growth of software products, while at the same time keeping day-today development work lightweight and structured. There also needs to be developed more structured ways to build and ensure trust; both in contracting and within projects.

Keywords: Software Estimation, Contractual Incompleteness, Agile Software Development, Planning Fallacy

Acknowledgements: Thanks to Michael Cusumano, Pierre Azoulay, Magne Jørgensen and most of all Arild Aspelund for discussions, input, resistance and encouragement along the way. Thanks to my employer for access to these interesting projects and to my colleagues for great conversations and input along the way; Hanne, Jørn Ola and Kristoffer especially. And finally thanks to family and friends for the perfect mix of support and distraction.

1 Introduction

1.1 Background

We tend to think of uncertainty as a negative term, and in our daily lives we try to reduce uncertainty. One would think that was even truer in a business context. And for the most part it is.

Contracts can be seen as means to minimize uncertainty and ambiguity. The question I would like to explore in this thesis is whether there can be contexts, environments and relationships where uncertainty could be considered a virtue, not a vice; a feature not a bug. Is contractual incompleteness something to be limited or can it give a competitive advantage in some contexts?

How can businesses in highly volatile, uncertain and complex domains - dealing with rapid changes both in technology and markets - strike the balance between responding to change and having a sufficient degree of certainty, predictability and control?

An example of such a complex and volatile domain is software development. In this thesis I investigate in what circumstances deliberately making contracts less complete and more ambiguous may be fruitful. I also explore how deliberate contractual incompleteness influences software development projects. How do more ambiguity and less control (in the classical sense) influence how we work and how we deal with planning, cost, value and product quality? These two questions also pose a third question; if we are leaving traditional contractual completeness behind, what new practices will emerge? Which phenomena are occurring organically and which mechanisms should be developed deliberately and systematically?

To draw upon Oliver Williamson (2005), it can be asked whether a high level of contractual incompleteness can be a wedge between the high transaction cost of the open market and the cost and risk of vertical integration. If a buyer in effect only contracts a number of resources with a certain competency and a high level "mission statement" as contractual clauses, is that both lowering the transaction cost of dealing in an open market, guaranteeing trade while at the same time reducing the negative effects of ex post contractual behavior such as shading and adverse selection?

1.2 Practical issues and implications – a managerial dilemma

Leaving contracts deliberately incomplete poses a managerial dilemma. A buyer needs to have a solid basis to make investments decisions. Common sense tells us that it is irresponsible to make large investments (or even a small ones) without knowing what they will cost you. It is indeed counterintuitive. So how may this still be the case?

An example of deliberate contractual incompleteness is leaving specifications and plans more vague than is possible. There are several reasons why planning in general, and estimating in particular is a more uncertain practice than we would like to think. In all kinds of planning and thinking about the future, a large range of cognitive biases limits us. In domains with rapid changes, detailed plans will also have a short timespan of validity. Before the plan can be implemented, technology, market, strategy and other factors may have shifted significantly.

In highly complex domains unforeseen issues often arise during development, this is addressed both in research on new product development (Brun & Gjelsvik, 2008) and in software development (Jørgensen, 2004).

Software is product where it is difficult for the buyer to assess quality ex ante – and even ex post. Software can be said to have "experience" attributes, meaning that their quality is assessable only after it is delivered (Mishra, Heide, & Cort, 1998). This may lead to opportunistic behavior on the part of the supplier - and fear of opportunistic behavior on part of the buyer, often referred to as adverse selection and moral hazard problems. Incentives on part of the supplier and signaling to reveal private information are often proposed to overcome these problems.

In the case of software development, one has become accustomed to using estimates as proxy for quality or value creation as control mechanism. But in fact controlling the accuracy of estimates has little or nothing to do with product quality. This is not a problem with estimates as such, but with how they are being used. What you get then is a target that is arbitrary to business value and quality. It means that governing a project by estimates may keep you running fast – but it might very well be in the wrong direction.

We know that we tend to do what gets measured. And if the measure is unrelated to what is creating value, then counterproductive behavior might arise.

An additional problem with estimates is that creating reasonably accurate estimates is costly. And if you consider that estimates are at the best of times about 30% off, then the cost seems unreasonably high.

Given the argument above, it seems understandable why Project Managers (PMs) would try to avoid both making detailed estimates and using them as proxy for

value and quality. Nevertheless I would like to investigate if there are biases working in the other direction as well. Are there less rational reasons why PMs would want to skip estimates? Because let's face it; estimating is a dreary job. And PMs must face the consequences when estimates prove to be inaccurate – which they always are. Estimating and tracking estimates is not a fun job by any account, it feels very liberating to lose that practice. "Just delivering value" feels so much more rewarding. Might this lead them to throwing out the baby with the bathwater when rejecting estimates altogether?

1.3 Theoretical problem

There are many theoretical fields that discuss the dilemma of ex ante uncertainty and the need for ex ante control and predictability. This is a topic is explored in software engineering research, in research on product development and more generally as a problem of corporate governance.

Research on software development and software cost estimation point to that estimates are fairly inaccurate. Studies show that achieving a higher degree of accuracy requires systematic, costly and rigorous practices. Software is also a type of product where quality is difficult to assess ex ante and even ex post. Harris et al. (Harris, Collins, & Hevner, 2009) addresses this dilemma using dynamic capabilities and agency theory to bridge the gap.

Research on product development and more generic research on domains with high levels of ex ante uncertainty, complexity, volatility and creativity explores whether detailed planning and estimating is less relevant and even counterproductive in that it hinders creativity.

Another view on how contractual relations play out and influence the transaction cost (particularly ex post), is the psychological or relational view on transactions. The term psychological contract is a relevant theoretical lens here, as are theories on bounded rationality and cognitive biases. Several cognitive biases are also limiting our ability to make claims about the future.

Transaction Cost Economics looks at the make-or-buy decision and prescribe integration when complexity and asset specificity becomes high. Williamson (2005) names specificity, uncertainty and frequency as three critical dimensions describing transactions. When all three are present to a large degree, integration has traditionally been seen as the best option.

There is plenty of research pointing towards less rigid plans as certainty decreases. But how then to maintain some degree of control even with more flexible and incomplete plans? That is a central dilemma that some recent research has discussed and I'll discuss this further in chapter 2.

1.4 Research questions

There are three questions I would like to investigate in this thesis. They are:

1. What are the contexts, environments and circumstances in which contractual incompleteness, in the form of not estimating or up-front planning, could be an advantage?
2. How does an incomplete contract affect the projects, the cooperation, processes and value created?
3. What practices and control mechanisms exist and can be put in place of estimates in order to overcome the dilemma of flexibility versus control?

2 Theory

2.1 Introduction

This chapter sets the theoretical stage for the study of contracting, planning and control in software development projects. I will draw upon two major research fields; transaction cost economics and cognitive bias theory. I will also look at research on software engineering and theory on customer co-creation and information stickiness in knowledge intensive business services to set a relevant theoretical background for this thesis.

The two main research fields have different starting points, but seem to point in the same direction and ask the same questions. Whereas research on cognitive biases that influence planning point to that predictions about the future is difficult and costly at best, the research on transaction cost economics (and specifically) contractual incompleteness suggests that in uncertain, complex, volatile and creative settings, it might be better not to make very rigid plans, or contracts.

The fact that contractual completeness is difficult, cumbersome and costly should, however, not lead us to the conclusion that we should abandon the idea. But as transaction cost theory and theory of incomplete contracts describes this as a positive factor makes the case for looking at other ways to conduct planning, contracting and control in complex, flexible and volatile domains where creativity is a desired capability.

2.2 Definitions

2.2.1 Software development

In this thesis I will consider software development as development of custom-made software made to a buyer's specifications. Since the buyer has deemed that simple off-the-shelf software is not the best fit, there tends to be a high degree of complexity and uncertainty. As custom development often deals with an unknown territory both in terms of technology, domain and market, there is considerable need for creativity in the process.

There are often multiple stakeholders, both internally in the organization and externally (customers and participants along the value chain). This can increase volatility in requirements and priorities. From a project point of view, this constitutes what contract theory would call market uncertainty.

Software development being in its nature a high tech endeavor, technological uncertainty and changes occur more often than in other sectors. Another aspect of software development is that it is still considered an immature business and market.

2.2.2 Estimates and estimating

There are many ways in which a contract can be incomplete. For the purpose of this study, I will look at contractual incompleteness as absence of estimates and to some degree absence of up-front planning and specifications. The reason why estimates are reasonable objects of study, is that they exists within the project's life cycle and they are possible to study without following a project from its inception to its end. Another reason why estimates are interesting objects of study, is that there is a lot of debate in the software development community on whether or not to estimate. Going forward, I will refer to contractual incompleteness and low levels, or absence, of estimating and up-front planning interchangeably.

In this thesis, I will refer to estimates as detailed, up-front software development effort estimates done by software development experts. The estimates usually try to quantify the effort needed to fulfill a set of requirements. Expert estimation practices often actively tries to uncover any hidden complexity, uncertainty or assumptions.

It will be instructive going forward to separate estimates and estimation practices from how they are being used. We can divide the uses into five categories; bidding, investment analysis (business case), project planning (resource and scheduling), prioritization and project control. Estimates are sometimes the only measure and control of project progress and success, a proxy for other more intangible measures and the only means by which we prioritize what to make.

2.3 Transactions cost economics

Transaction costs are the costs that are incurred when entering into a transaction; contracting, risk, selection, integration etc. Transaction cost economics is able to explain why not only price, availability and demand rule the market alone; there is cost and risk associated with entering into business with a new partner. The (neo-) classical Smithsonian economic theory did not account for this.

TCE has shed light on the make-or-buy decision, investigating what makes companies choose how to go to the market; by contracting or by integrating. The choice between market and hierarchy is another common way to pose the same consideration. The choice of transaction method is closely linked to the degree of control the buyer can and need to have. Making a "simple" purchase in the market gives little control, but has a low transaction cost. A complete vertical integration on the other end of the spectrum, gives complete control, but also poses considerable risk and has a larger cost.

Another central topic in TCE is what kinds of interactions with the market is preferable depending on whether there is likely to be ex ante and/or ex post changes and negotiation. Traditionally, a high likelihood of ex post negotiation has been seen as a path to vertical integration as it is difficult to describe and decide the exact details of the contract prior to contracting, ex ante.

Anderson and Gatignon point to four central aspects of Transaction Cost Economics that determine how the firm should organize its transactions (E. Anderson & Gatignon, 1986):

- Asset specificity; the degree to which the parties need to build or develop assets that are specific to the contract/relationship and that could not (entirely) be recouped if exiting the contract/relationship.
- Internal uncertainty refers to when the buyer cannot ascertain seller's outputs, the quality of the services delivered, also called experience attributes. This is related to the risk of opportunism; to what extent does private information on part of the supplier put the buyer at a risk of opportunism. This is a relevant scenario in purchasing knowledge intensive services such as software development services. The buyer will need to either monitor inputs or use more subtle incentives to align goals and increase loyalty.
- External uncertainty refers to volatility of the firm's environment. This is what I will refer to as market uncertainty and technology uncertainty.
- Free-riding potential: the agent's possibility to receive benefits without bearing the costs. This is not the most likely scenario in software development, but it could maybe be parallel to not delivering the quality of competency that was specified.

The Fundamental Transformation is a term used by Williamson (2005) to describe the change in relationship, attitudes and dependence between two contracting parties before and after a contract is entered into. Before contracting, the relationship between the negotiating parties is often one of competition, distrust and distance. In regulations of public procurement, this distance is even regulated by law; there is to be very limited contact between the offering and the procuring parties before the contracting is finalized in order to guarantee neutrality and fair trade. After contracting on the other hand, the relation changes fundamentally and is often described as a bilateral relationship or even as a bilateral monopoly (Caballero & Hammour, 1996). This is especially due to the existence of contract specific assets the parties have or will have to invest in. The idea is that both parties will have something to loose by breaking out of the relationship. On one hand, this may bring a sense of loyalty and alignment to the relationship, but it may also be a counterproductive mechanism in that it creates a lock-in effect and deters change and creation. Another aspect of the Fundamental Transformation this thesis touches upon, is whether a bilateral relationship with a high level of trust and common conceptions of the overall goals and vision can be a good environment for co-creation in contexts with high uncertainty and volatility.

2.3.1 Contractual incompleteness

The term Contractual Incompleteness stems from the TCE. Contractual incompleteness has traditionally been seen as a negative fact of contracts in the real world; a contract cannot possibly describe all possible contingencies and outcomes of a contract. The reasons most commonly used to explain contractual incompleteness are transactions cost and bounded rationality; it would be too costly to describe all contingencies in the real world, and imperfect information and cognition would limit what contingencies we would be able to envision. Several researchers point to that these two phenomena alone cannot explain incompleteness of contracts (Fehr, Hart, & Zehnder, 2011; Halonen-Akatwijuka & Hart, 2013). In this chapter I will look into findings that explains and nuances the concept of contractual incompleteness.

Bajari and Tadelis explore different contract types and their adaptability to ex post changes as a source of, or explanation for, contractual incompleteness:

> *"In fact, Williamson expresses the idea that low incentives are good to accommodate ex post adaptations and writes (1985, p. 140) "low powered incentives have well-known adaptability advantages. That, after all, is what commends cost-plus contracting. But, such advantages are not had without cost-which explains why cost-plus contracting is embraced reluctantly."* (Bajari & Tadelis, 2001)

Whereas fixed price contracts (that have high incentives) may lead to costly ex post adaptability partly due to private information on part of the supplier, low incentive

contracts (Cost Plus, which in software development often take the form of Time and Material (T&M) contracts) have little incentive to be cost effective, but can accommodate ex post changes more readily. This is particularly relevant for contexts with a high level of uncertainty both regarding market and technology since ex post changes are likely to occur. Bajari and Tadelis (2010) conclude: *"Our analysis suggests that if the likelihood of changes to a design is large, then the buyer should choose weak incentives"*

Halonen-Akatwijuka and Hart (2013) investigate other motives for leaving (more or less deliberately) contracts incomplete; to prevent shading and disagreement about ex post contingencies where the parties does not have the same view of "the state of the world".

What the abovementioned studies have in common, is that they look to ex post situations to explore and explain contractual incompleteness:

> *"These observations suggest that the procurement problem is primarily one of ex post adaptations rather than ex ante screening."* (Bajari & Tadelis, 2001)

The research on incomplete contracts, transaction cost theory and IT outsourcing (Bahli & Rivard, 2003; Fink, Lichtenstein, & Wyss, 2013; Lacity, Willcocks, & Khan, 2011; Susarla, Barua, & Whinston, 2009) tends to take the contract type as object of study. There are clearly many ways in which contracts can be incomplete. For the purpose of this thesis I will look at low levels of up-front planning, specifications and estimates as contractual incompleteness. My proposition is that the contract type itself is not as central to the problem of contractual completeness as the completeness of the content of the contract - of the specifications and the estimates. It's the contents and understanding of specifications and estimates that are at the core of ex ante negotiations and ex post renegotiated through more or less rigorous change management procedures. If changes to specifications and estimates are probable due to uncertainty, volatility or complexity, then an incomplete contract with a better possibility for ex post renegotiation may, according to theory on contractual incompleteness, be more appropriate.

2.3.2 Contractual incompleteness and uncertainty

In their study of software procurement in a large international bank, Fink, Lichtenstein and Wyss' research "…*suggest that project uncertainty is negatively associated with contractual completeness*" (Fink et al., 2013), that is; the higher the uncertainty, the more incomplete the contract is likely to be.

Bernheim and Whinston put a normative dimension to it:

> *"…we argue that such incompleteness is often an essential feature of a well-designed contract. Specifically, once some aspects of performance are unverifiable, it is often optimal*

> *to leave other verifiable aspects of performance unspecified"* (Bernheim & Whinston, 1998)

Verganti and MacCormack look along the same lines and identify context or environmental factors as deciding for what process is best associated with performance (MacCormack & Verganti, 2003). The contexts they investigate are the levels and sources of uncertainty; market and platform uncertainty. Platform uncertainty in our context would translate to market and technology uncertainty. What their study shows, is that flexible processes in these contexts are associated with better performance. Fehr, Hart and Zehnder's research point in in the same direction:

> *"Such enhanced flexible contracts would be attractive as they would not only guarantee trade, but would also avoid inefficiencies caused by aggrievement and shading*." (Fehr et al., 2011)

Eisenhardt and Tabrizi (1995) suggest that maturity of business is deciding for what kind of development processes that are associated with development effectiveness. This is relevant to this thesis in so much as software engineering is considered a business with low maturity (as discussed in 2.2.2).

These studies are in line with my first research question: under what circumstances can software development contracts be left less complete?

Another aspect that is relevant for contractual incompleteness is the fear of opportunism on part of the supplier. Fink et al (2013) link volatility, uncertainty and chance of opportunism linearly to contractual completeness:

> *"High specification volatility projects, for which complete contracts would be costly to draft, are likely to be based on the less complete archetypes. Similarly, these incomplete contracts are likely to be granted to familiar vendors for whom the likelihood of behaving opportunistically is low."* (Fink et al., 2013)

The correlation between fear of opportunism and familiar vendors, or even reputation in a market may also prove relevant here. It certainly is likely that a buyer will be less afraid of venturing into an incomplete contract with a supplier that is either well known to them directly (through an existing contract or relation) or that has a widely recognized good reputation in the market.

I will look at estimates and their levels of ex ante detail as a proxy for contractual completeness. In Norwegian software contracting there are often large and spanning framework agreements that are quite general in terms of specifying procurement scope. For each project (the object of my study), a more specific sub-contract is written. This sub-contract may or may not be complete in terms of specifying in detail what is to be produced. It is at this point it's most commonly

decided how complete or detailed the contract is to be – and the existence and use of the estimates provided are at the heart of that question.

2.3.3 Contractual incompleteness and ambiguity

Though there is not a lot of academic research on contractual incompleteness in software engineering, there are some interesting studies on this topic in the field of New Product Development. New Product Development has many commonalities with software development and research on this field is useful and relevant for the study of software development:

> *" Increasingly, this correspondence is direct because many development teams work on software products, including shrink-wrapped software, enterprise resource planning systems, and Internet-based services. Even custom software development has similarities with product development "* (Harris et al., 2009)

In product and software development alike ex post change is traditionally seen as something to be limited or at least controlled rigorously. Change management is a large, costly and bureaucratic process in Software Engineering. In the gold standard of project management literature, PMBoK (the *Project Management Book of Knowledge*), the process of Integrated Change Management is the largest, most spanning and most complex process.

However recent research challenges this assumption of late change as a problem. It is posited that keeping options and scope deliberately open, can increase performance by allowing creativity in more phases of the development process than just the classical "Design and specifications"-phase (Brun & Gjelsvik, 2008) (Tabrizi & Eisenhardt, 1995).

Brun et al. make an interesting distinction between uncertainty and ambiguity. Whereas uncertainty is considered lack of information, ambiguity is defined as *"different interpretations of the same piece of information"* (Brun & Gjelsvik, 2008). They found four categories of benefits from sustaining ambiguity: Saving cost, saving time, retaining fallback options and retaining product ideas.

Deferring decision and keeping options open are values often mentioned in writings on Lean startup and agile methodologies; releasing software in a "Minimal Viable Product" (Ries, 2011) in order to get feedback and pivot and making decisions at the "Latest responsible moment" (Poppendieck & Poppendieck, 2013).

In the context of this study, I will look at the level of uncertainty and ambiguity and its correlation to a high degree of flexibility in the process. As shown by Verganti and MacCormack, flexibility is associated with better performance.

Building on that finding, I'll look at what circumstances enable a higher level of flexibility – or a lower level of up front plans, specificity and estimates.

2.3.4 Psychological contracts

A psychological contract is defined as *"an individual's mental beliefs about his or her mutual obligations in a contractual relationship"* (Rousseau, 1989). Rousseau deals primarily with employment contracts in her research. Koh, Ang and Straub (2004) bring the aspect of psychological contracts into the realm of IT, more specifically IT outsourcing. They emphasize three aspect of psychological contracts in their study on psychological contracts in IT outsourcing; mutuality, the existence of psychological obligations and the fact that psychological contracts exist between individuals (Koh, Ang, & Straub, 2004). They have focused their study on the relation between buyer and supplier Project Managers. They are taken to be agents expressing their firm's views and interests; they are largely responsible for facilitating and assessing the relation.

Their study found a positive correlation between outsourcing success and fulfillment of the obligations of the psychological contracts. They also made the surprising finding that actual product delivery was not among the top 6 things the parties expected of each other.

A meta-study conducted by Lacity, Willcocks and Kahn (2011) looks beyond TCE theory and especially propose that certain supplier and buyer properties determine project success; maturity and distinctive capabilities on the supplier side and prior experience (with IT Outsourcing) on the buyer side. This matches the perspective of psychological contracts, as the agents are the foremost representatives of their company's capabilities and culture.

I suspect that psychological contracts play a larger role in cases where contracts are incomplete; trust, relations and reputation come into play when there is less complete contracts to lean on. A good relation will also work as an insurance against shading and opportunism.

2.4 Dynamic capabilities and agency theory

Resource based theory combined with agency theory offers a lens to look at the degree of flexibility in processes. Dynamic capabilities could shed light on how organizations perform and behave when a more flexible process is needed; in contexts with high market uncertainty and technological innovation and volatility. Agency theory could explain how a more flexible process could still afford the principal with a level of control and predictability.

Harris et al. defines dynamic capabilities as:

"…the ability to reconfigure resources by effectively sensing the environment, learning, coordinating, and integrating interaction patterns" (Harris et al., 2009). [1]

Dynamic capabilities has a less static view of the firm (and for our context, the project) than transaction cost theory. Eisenhardt and Tabrizi (1995) argue that classical TCE is not well suited to explain organizations at high speed and high level of uncertainty.

Harris et al. validates the use of dynamic capabilities onto software development in two ways; first through the similarities between software development and product development and second more logically; software projects are indeed often influenced by market uncertainty, but also shifts in "internal" markets; organizational changes, multiple and shifting stakeholders and changes in goals and priorities. Technological uncertainty is indeed, as discussed earlier, a dominant feature of software development.

Agency theory deals with principal–agent relationship. In our context, the principal would be the buyer and the agent the supplier in a software development contract. How can the principal assure that the agent acts according to her intentions and towards her goals? Contracts and incentives are obvious candidates here, but as discussed above, high incentives and complete contracts are less ideal in situations where there is uncertainty. Further complexity is added in cases where quality is difficult to ascertain ex ante or even ex post, what TCE refers to as internal uncertainty. As already discussed, software is a kind of product where quality is difficult to verify even ex post and both market and technology uncertainty is high.

The question these two theoretical lenses pose, is how to create value from dynamic capabilities while allowing the principal some way to assure that the agent is in line with her goals.

---

[1] On a side note, this could also be a definition on how a well-functioning consultancy would have to work.

Harris et al. (2009) offer a model to encompass these two aspects. Frequent feedback is suggested as a way to give the principal some degree of control. This can limit moral hazard and adverse selection. A varying degree of "scripting" or ex ante detail is suggested as a lever to develop and utilize dynamic capabilities. Harris et al. draw a matrix to illustrate levels of process flexibility and control.

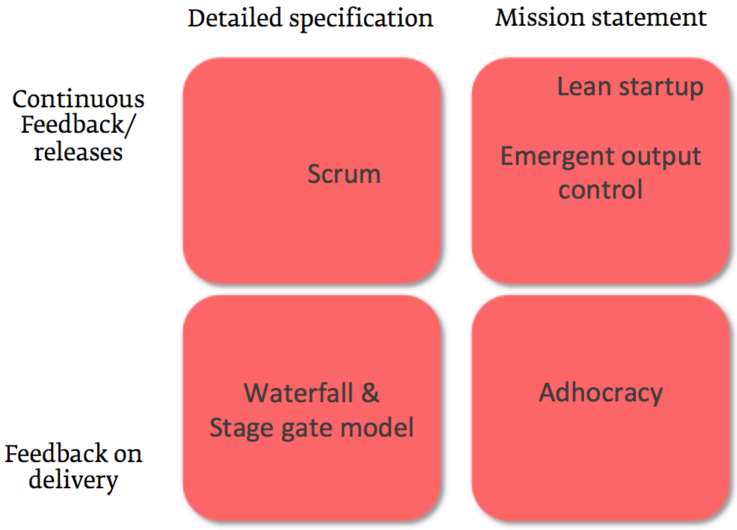

Figure 1 Model describing the relationship between control and flexibility in software development, inspired by Harris et al. (2009)

Harris et al. uses the analogy of improvisation and creativity. The level of improvisation is a useful metaphor for explaining how development teams respond to different levels of scope specificity. Their study support the hypothesis that higher market and technology uncertainty is more likely to be managed by a more flexible process. It also seems to support that success (defined as matching market needs) in uncertain projects is more likely with a controlled-flexible process (than with plan-driven and ad hoc methods). Their findings are thus in line with other research on volatile, ambiguous and uncertain domains discussed in chapter 2.3.2.

Eisenhardt and Tabrizi's study from 1995 point in the same direction; with high uncertainty, iterative processes and testing with the possibilities they give for evaluation, learning and building confidence are associated with fast product development (Tabrizi & Eisenhardt, 1995).

What is fruitful in these studies is the exploration of possible control mechanisms in flexible processes – the continuous feedback or emergent outcome control in their terminology. This has similarities with the current Lean Startup trend; fail early and often, or in agile software development terms; inspect and adapt.

2.5 Cognitive biases and organization theory

Detailed specifications are a prerequisite, if not the core, of traditional, complete software development contracts. The buyer needs to have an idea of what they want and the supplier needs to be able to tell what it will cost to build. Both creating specifications and estimating the cost of responding to them, is about predicting a future state of the world. Even the most detailed specifications have implicit assumptions in them. And estimating involves a lot of guesswork and choices of how to solve the requirements. This is especially true if the domain is foreign to the supplier and the contact between the parties is limited. On both sides of the table, there is made a lot of assumptions about the future.

There is plenty of research on what limitations and inclinations we are subject to when making predictions about the future. Among the most influential are the Nobel Prize winning Kahneman and Tversky (Tversky & Kahneman, 1974), (Kahneman, 2011).

The most relevant cognitive biases for my research objectives are planning fallacy and intuitive statistic fallacy, attribution error, selection bias, priming and anchoring.

2.5.1 Planning fallacy

The planning fallacy refers to the tendency people have to underestimate how long it will take to complete a task. Research shows that even when we are aware of this cognitive bias and take it into account, we are still under its spell. Hofstadter's law speaks to this very explicitly: *"It always takes longer than you expect, even when you take into account Hofstadter's Law"* (Hofstadter, 1979)

But how about professionals with long experience in estimating and have everything to gain from being accurate? Could we put our trust in professionalism and the assumption that software projects are estimated and lead by people who have a wish for control? According to Halkjelsvik, Rognaldsen and Teigen, this may be the wrong bet:

> *"People with a desire for control produce too optimistic time estimates when rewarded for fast performance. It follows, somewhat paradoxically, that their desire for control can be self-defeating, by interfering with their ability to set realistic goals and formulate feasible plans. They may also experience more impatience and suffer stronger disappointments. Thus, in a sense, their need of being in control may prevent them from gaining control."*
> (Halkjelsvik, Rognaldsen, & Teigen, 2012)

2.5.2 Attribution error and Availability bias

Attribution error refers to our tendency to ascribe positive outcomes to our own good work and skills, and negative outcomes to external factors. We also tend to

discount the role of luck in positive outcomes. This inhibits us from predicting on the basis of past experience.

A related bias is the bias of availability and substitution. When asked to estimate about size or frequency, we tend to substitute the question with an easier one; the impression or event that is nearest to the top of our minds; "*report an impressions of the ease with which instances come to mind*" (Kahneman, 2011). Attributes such as familiarity, salience and recentness influence what comes to mind easy.

2.5.3 Anchoring and Priming

Anchoring refers to the tendency to be influenced by numbers (or other facts) that are presented to us prior to making judgments, but that are completely irrelevant to what we are making judgments about. Even if we know we are presented with irrelevant information, we are still anchored to it. In bidding there are lots of numbers that can be anchored upon; general information about the value of the firm in question, the last project you delivered, an assumption about the buyers expected price. And professionals seem to be just as susceptible to anchoring as people who have no subject matter knowledge:

> *"The only difference between the two groups was that the students conceded that they were influenced by the anchor, while the professionals denied the influence"* (Kahneman, 2011).

Priming is similar type of involuntary and unconscious mechanism that influences our ability to make judgments about the future. We can be primed by any number of things; from how our general mood is on that particular day, if we feel (or are made to feel) powerful, and information that is not at all relevant to the task at hand.

Not only do we have bounded rationality, our minds constantly try to compensate for the information we do not have. A mechanism that hinders good judgment on matters where we have little information is what Kahneman (2011) calls *What You See Is All There Is* - WYSIATI. It is linked to both overconfidence bias and confirmation bias, we tend to be very quick about making judgments even though we have very little – and completely irrelevant – information:

> *"Paradoxically, it is easier to construct a coherent story when you know little, then there are fewer pieces to fit into the puzzle. Our comforting conviction that the world makes sense rests on a secure foundation: our almost unlimited ability to ignore our ignorance"*
> (Kahneman, 2011)

This has several interesting impacts on the topic of this study. If we make plans, specifications and estimates at a very early stage, we probably now very little of the complexity and depth of what we are to make. But we overlook what we don't know, the WYSIATI is at work. And as we gradually learn more of the topic at

hand, confirmation bias has the tendency to hold us to decisions we made before we had all this valuable, new information. This in turn hinders us from exploiting the possible ambiguity and creative space that would have been there had we not made detailed plans and specs up front, as discussed by Brun and Gjelsvik (2008).

2.5.4 Selection bias

Selection bias is a statistical bias that can lead us to make skewed selections from a sample. Imagine a department head wants to get a project approved and funded in her organization. She is probably the one who is responsible for providing the budget and estimates on what the project will cost (and the returns it will yield). She will have strong incentives to make the investment look as promising as possible. This will reinforce the planning fallacy we all are subject to even if we don't have any stakes in the outcome. And overoptimistic plans have a higher chance of getting selected, as they appear to be better investments.

Selection bias is one of the reason why surveys show such large cost overruns in software development projects; the lowest bidder is more likely to get chosen and also at the same time more likely to have underestimated the cost of development (Jørgensen, 2013).

2.5.5 "What gets measured gets done"

It is a common managerial dilemma of wanting to measure key performance parameters without the risk of getting counterproductive behavior as a result. This is also the risk of having a metric such as effort estimates be a performance indicator. You stand at the risk of having teams trying to fulfill estimates at the expense of the real values that are far more difficult to measure, but that the customer actually wishes to attain.

> *"…contracts that tie an agent's compensation to verifiable measures of performance can divert effort and attention from other more important, but less easily measured, aspects of performance" (*Bernheim & Whinston, 1998)

A related mechanism is that plan, estimates and schedules tend to be self-fulfilling or normative. Several studies on the effects of software estimation and scheduling show the normative effect of estimates. Projects are prone to fit the work to the estimates – both if the estimates are overly optimistic and pessimistic (Abdel-Hamid & Madnick, 1986; Jørgensen & Sjøberg, 2001).

2.6 Customer co-creation and information stickiness

Knowledge intensive services such as consulting and especially custom-made software has a high level of collaboration and co-creation aspects. The buyer has a problem that cannot be solved simply by procuring off-the-shelf software. In most cases it requires the software supplier to go far into the customers domain and the customer to understand a lot more of the technology involved then they expected to. The two parties need to build a common language in which to discuss and elaborate the solution. Both parties partake in the creation of the software, thus co-creation.

Another element of customer/supplier co-creation is the sticky nature of information (Hippel, 1994). In order for knowledge to remain in the organization, there needs to be real participation from both sides. Information cannot simply be handed over like a material asset or even in form of documentation. Hippel speaks of an information transfer cost.

Xue and Field (2008) explore the intersection of customer co-creation, information stickiness and incomplete contracts in knowledge intensive business services: "*we identify and model contract incompleteness as a relevant and yet unexplored source of information stickiness*" (Xue & Field, 2008). They propose that incomplete contracts increase information stickiness and thus the cost of information transfer. Another way to look at it would be to say that both information stickiness and contractual incompleteness are consequences of procurements with a high level of complexity, uncertainty and ambiguity. In their paper Xue and Field take contractual incompleteness to mean how complete the specification of division of labor and responsibility is between the consultant and the client, which is slightly different from the specific type of contractual completeness this thesis investigates. Nevertheless, closeness to and availability of information transfer between client and consultant is a crucial part of consultancy practices and contracts.

2.7 Software engineering research

There is a lot of academic research on software development, but as always, academic research is lagging slightly behind current practices. For this reason, there is growing amount of research on research on agile and especially SCRUM software development methodologies, but little academic research on the current trends in software development; methodologies inspired by Lean manufacturing and Kanban. These methodologies quite often entail removing estimates and fixed iterations. What is lacking in academic research is not lacking in publishing. The books, conferences, articles and blogs on the topic of lean software development is indeed not lacking in volume, but is mostly opinion based and anecdotal.

What I have been investigating the previous chapters, is if there is a basis for the current practices and SE folklore in academic research on other fields and businesses. Now it's time to turn to current practices and theory on software development, both as a theoretical backdrop, but also to understand the context of the empirical research.

2.7.1 Agile Software Development – From Agile to Lean

The Agile Movement started as a reaction to the increasingly rigid software development methodologies of the 80-ies, especially RUP, and their lack of the promised predictability. More lightweight project methodologies started emerging. In line with the current thinking of Lean manufacturing sprung the Agile Manifesto in 2001 (Beck, Cockburn, Martin, & Schwaber, 2001). The manifesto declares:

"Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan"

The biggest change from the classical waterfall (and stage gate) methods to agile methods are the focus on iterative development cycles in order to enhance learning and feedback and also the accommodation of change as part of the method. Difference in levels of up-front planning, documentation and closed gates between the different phases is especially relevant for this thesis.

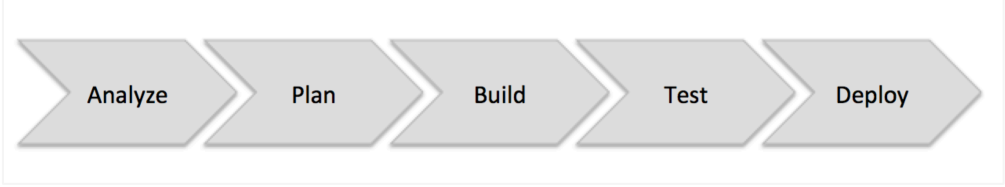

Figure 2 Illustration of a Waterfall software development process

The agile movement has made an enormous impact and has now; some 12 years later become the de facto standard for software development methodology in Europe and the US. But whereas practices change fast, procurement- and legal processes are slower to adapt. There are, however, two new standard contracts for software development in Norway that are more in line with more agile and less formal governance.

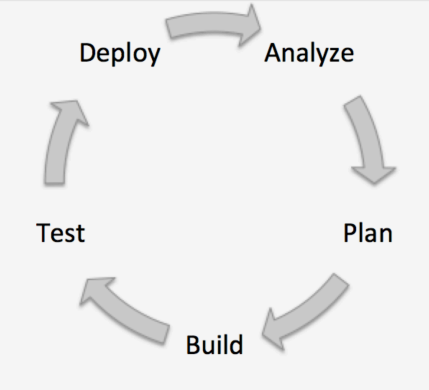

Figure 3 Illustration of an agile, iterative software development process

A criticism of agile software methodology, SCRUM and xP especially, is that it's very developer-centric and supplier friendly. That may be the case, but on the other hand both suppliers and buyers of software acknowledged that the "traditional" way of procuring and developing software was flawed and that something needed to change. The acceptance of these more lightweight methods is now largely ubiquitous.

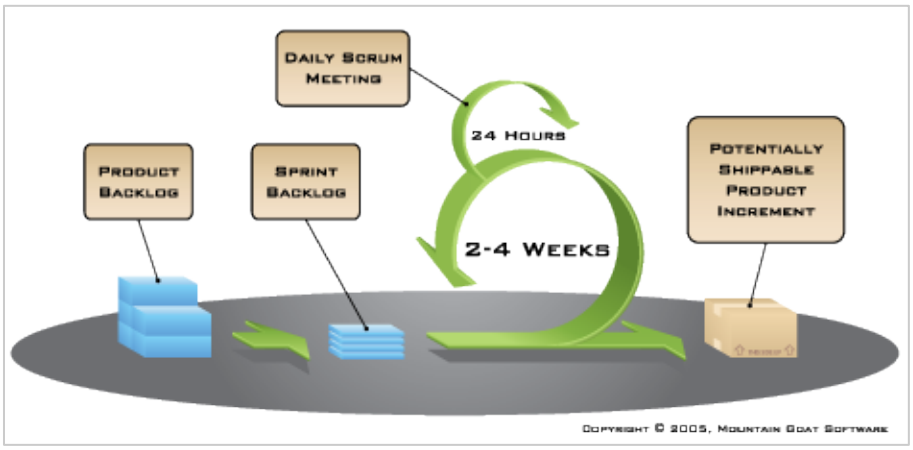

Figure 4 Mike Cohn's model of the SCRUM process (Cohn, 2012).

In recent years another influence from Lean manufacturing has made great impact on software development methodology; Kanban. The biggest practical change from "vanilla" agile software development is the removal of fixed iterations and an

adaptation to pull based and Just In Time (JIT) scheduling system. The goal is to optimize for flow, minimize queues and remove impediments in the process. One tracks cycle time and average lead-time to predict delivery. Another prominent feature of Kanban is the use of workflow visualization. Kanban is indeed Japanese for signboard or billboard. The Kanban board is crucial for illustrating status of the work in progress and make clogs and bottlenecks visible.

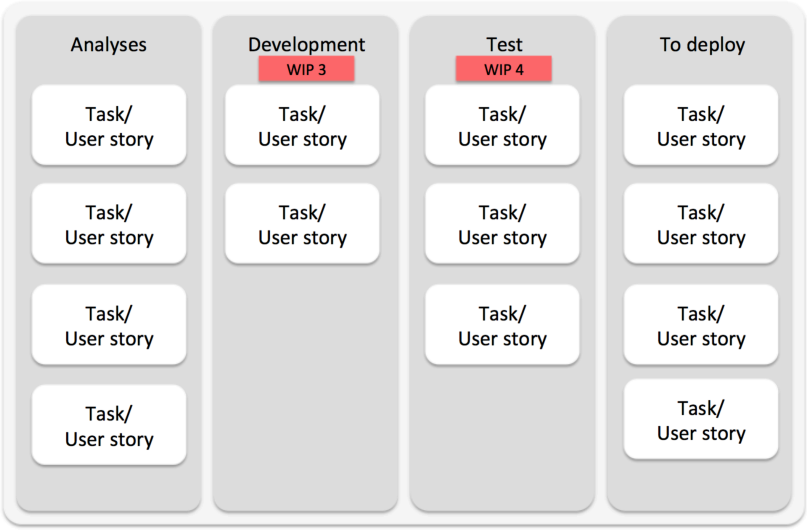

Figure 5 Simplified model of a Kanban board in a Software Development Project. The workflow steps in view here are Analyses, Development, Test and (ready) to deploy. The yellow WIP-signs indicate that there is a limit to how many tasks that can be at a workflow step at a time – Work In Progress.

For the scope of this thesis, the removal of iteration- and release planning sessions are particularly relevant. It is in the release and iteration planning sessions that estimation was done by the entire team in SCRUM projects.

Lean Startup is another trend that has reached and influences software development practices the recent years. The core of Eric Ries Lean Startup-model is the "build, measure, learn"-loop (Ries, 2011). It's not only from the measuring that you learn, but also from the building. As with many creative processes, the building or creating is in itself an exploratory and creative process.

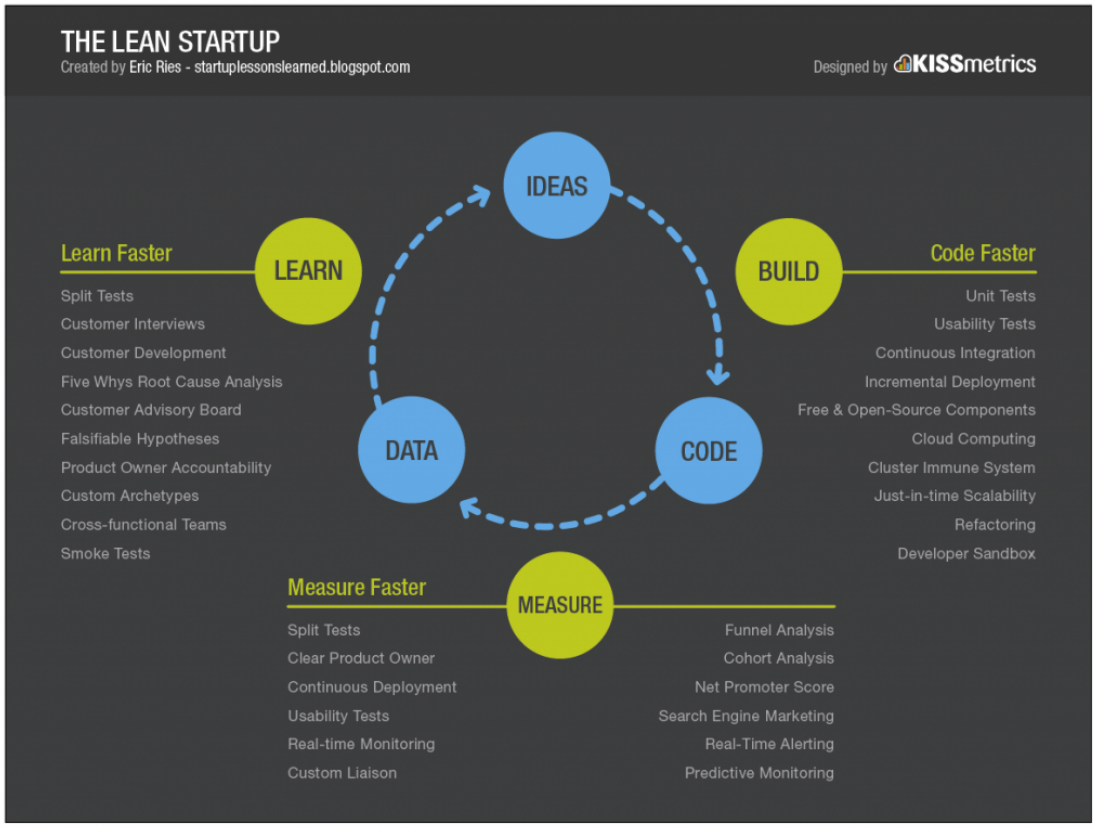

Figure 6 Illustration of The Lean Startup by Eric Ries.

Another relevant idea for this thesis is the idea of a Minimal Viable Product (MVP). The point of an MVP is to be able to learn, iterate and test sooner rather than later. It resonates of the Extreme Programming slogan "Do the simplest thing that could possibly work".

A multiple case study from 2011 looked at how team processes and team formation was impacted by agile practices (Stray, Moe, & Dingsøyr, 2011). Their question was; does less planning impact how the team communicates, plans and select or prioritize work? Their finding was that in as much as planning and overall scoping was done outside the team, this resulted in weaker team formation and that tasks were selected not according to the customers priorities, but rather depending on the individual developers preferences, a classic case of the agency problem discussed in chapter 2.4.

There is some research on the use of Lean methodologies and Kanban specifically in software development (Ikonen, Pirinen, Fagerholm, Kettunen, & Abrahamsson, 2011; Senapathi & Srinivasan, 2011), but the field is still young in terms of academic research.

There is a discussion in academia on whether Lean principles and practices made for manufacturing are at all a good match for product development. Missing the big picture is pointed to as one weak side of the methodology; another is the elimination of waste itself. If we agree that product development (and software development) are creative endeavors where ambiguity, variation and open-endedness are of great value, then a process which primary goal is waste reduction might not be the best fit. This discussion is beyond the scope of this thesis, but is nevertheless an interesting one.

2.7.2 Agile estimation

Discussions on estimate inaccuracy, cost overruns and non-estimating practices in software development beg the question: what is so different about software development? I would argue, and the literature on other uncertain and volatile businesses suggests, that software development is indeed not that different. It has these problems in common with other domains where volatility, uncertainty and creativity dominate.

Research on software development estimation show that software project cost overruns are large, one much cited report from the Standish Group in 1994 (Johnson & Yarmouth, 2001) claims cost overruns were on average 189%, whereas Jørgensen and Moløkken (2006) claim that the report is methodically flawed. Other peer reviewed research on software project cost overruns show an average ranging from 30% to 40 % (Jørgensen & Moløkken-Østvold, 2006). There is a lot of research, and research still to be done, on how to improve software development cost estimation, but that is beyond the scope this thesis.

A recent study made an experiment where the same developers were asked to estimate some development tasks in three rounds some weeks apart. Some of the tasks were recurring, some were new. When estimating the second and third time, they did not have access to the earlier estimates. The mean difference between estimates made by the same developer at different times, were as high as 71% (Grimstad & Jørgensen, 2007). This might be an example of several of the biases discussed in 2.5. and an example of how unreliable software estimates seem to be.

"*Estimate size, derive duration*" is one of the slogans of agile software estimation (Cohn, 2005). The goal is to tone down some of the most "harmful" side effects of software estimates and enhance some of the strengths. By estimating in size or story points or other "non-monetary" units, one tries to get away from estimates being used for control and punishment purposes in a direct way. This also enhances the exploratory and investigatory aspects by applying group estimation processes that are focused on unveiling complexity, assumptions and divergence in information about the task at hand. It is thus hoped to counteract over-optimism

and alternatives bias discussed above. In addition it has the effect of bringing the team into a common commitment.

In *The Business of Software* Michael Cusumano describes how Microsoft develops their "Sync and Stabilize" method for development of large software systems (a new version of Office) after an iterative model; *Focus on creativity by evolving features and "fixing" resources"* (Cusumano, 2004).

They worked virtually without detailed specifications and estimates, but with a clear vision and only a prioritized "outline feature specification":

> *"Microsoft project managers generally scheduled in a simpler way. (...) they want a new release in twelve months and they have 100 developers and 100 testers. The problem then becomes how many features can a team of 200 engineers build, debug, and stabilize in eleven months or so, with some time set aside as a buffer? If they have done their prioritization and design work correctly, they can cut features if the project falls behind and include them in the next release or not at all"* (Cusumano, 2004).

This is often described as a time box-scheduling model. An important distinction from the scenario Cusumano is describing and the type of projects this thesis investigates, is that Microsoft has a vertically integrated software development staff; they have chosen hierarchy over procurement in the market.

There is a lot of academic literature on software estimation (1 700 hits in Web of Science for topic "Software cost estimation"), but agile software estimation is still a small and new field academically. One would expect that current literature on software estimation would discuss the ramification for agile projects as well. This is beyond the scope of this thesis.

2.7.3 Non-estimation

It does seem counterintuitive to enter into any form of contract without knowing what it will cost you. An investment decision is hard to fathom without at least some indication of what you will have to invest. How does build a business case without having a number on the investment to be made? In the context of software development, how do you make an investment without having specification and estimates to go by?

In the agile software development literature, both the academic and the popular, estimation is a frequent topic. Several more lightweight estimation models and processes are described and studied. However, there is very little academic research on the topic of non-estimation in software development. The only mention I have been able to find, is in a paper from *The Journal of Systems and Software* from 1985; *Managing Programming Productivity* (Jeffery & Lawrence, 1985). A number of factors are investigated as possible sources of productivity. One of these factors is how

program effort is estimated. The results show a higher productivity associated with not estimating. Jeffery and Lawrence explains this finding:

> *"…may possibly be due to a tight deadline preventing estimation and increasing pressure on the programmer for rapid completion of the job, or simply that for easy tasks no estimate was made"* (Jeffery & Lawrence, 1985)

The authors might have knowledge that tell them that the lack of estimates in these cases were indeed not a deliberate choice, but the way I read the article, the possibility does not seem to have occurred to them. They look upon it as an anomaly. And in all fairness, it is a counterintuitive idea.

Harris et al. have also encountered some non-estimating/non-planning teams, but seem to attribute the occurrences to time pressure or other unwanted, critical situations (Harris et al., 2009).

Eisenhardt and Tabrizi come closer to seing non-planning as a deliberate choice as they find that "*decision makers avoid planning because it is a futile exercise when the environment is changing rapidly and unpredictably*" (Tabrizi & Eisenhardt, 1995).

In the blogosphere and at software industry conferences, there is much talk about not estimating, but it is anecdotal and opinion based. Among the proponents of are David Anderson (D. J. Anderson, 2013), Neil Killick (Killick, 2013) and Woody Zuill (Zuill, 2013).

3 Method

3.1 Developing the research question

In the ongoing debate about estimation or non-estimation in the software industry these days, there is a tendency to postulate maxims and absolute truths that removing estimates is the silver bullet. To look into and hopefully nuance those claims, I would like to investigate the contexts and environments in which projects unfold and see if it has some bearing on the need for, and usefulness of, estimates. Thus my first research question; under what, if any, circumstances could removing estimates be a virtue.

My second question is probably somewhat harder to get a very clear answer to; what does it do to a project to remove estimates? It seems obvious that removing estimates frees more time to do productive work, but does it have other effects as well?

I'm also looking to find what control mechanisms can come or has come in the place of estimates to overcome the managerial dilemma described in 0. How do projects that have a low degree of up-front planning deal with the need for predictability and control? Are there practices, methods and mechanisms that seem to be emerging? Do any of the projects miss something in their processes or ability to plan or control?

3.2 Research method

3.2.1 The unit of analysis

My unit of analysis will be the project.

The Project Management Institute (PMI) defines a project thus: "*A project is temporary in that it has a defined beginning and end in time, and therefore defined scope and resources. And a project is unique in that it is not a routine operation, but a specific set of operations designed to accomplish a singular goal.*" (PMI, 2013) There are especially two parts of that definition that are interesting to our endeavor here. The first is that a project tries to deal with a unique problem. That entails a high degree of problem solving, often with use of lateral thinking and creativity. It also implies a level of uncertainty.

The second interesting part of the definition is "defined scope and resources". It is interesting because it may not be true in the projects I'm researching here, at least not if "defined" is to be understood as detailed and planned scope. That gives us two options; either they are not projects at all or the definition is outdated and as such not valid. PMI tends to have a rather traditional view of projects, much in line with waterfall methodology, a large part of the PMI processes has to do with up-front planning and change management.

Another definition of project that might be useful is a less normative one. One could define a project by its organization; is the endeavor organized as a project within the customer organization, it can be defined as a project. All of the projects in my sample fall in under this definition, whereas the first definition is inconclusive with regards to project A.

I could have chosen the contract as the unit of analysis, but the same contract can be executed quite differently even within the contractual relationship (same Vendor and Buyer). This is further mandated by the fact that there is a tendency to have long spanning framework agreement, especially in the public sector, with a set of suppliers from which the buyer can have several projects.

In order to research the abstract term "the project", I have aimed at getting more than one perspective on the project. As shown in Jørgensen and Moløkken-Østvold (Magne Jørgensen & Moløkken-Østvold, 2004), who you ask is quite relevant. I have therefore selected projects where I had access to more than one point of view; typically both buyer and supplier Project managers and a developer. I have also had access to observe the team at work. This allows triangulation of viewpoints.

The investigation will be comparative in nature, both in terms of points of view, but also in terms of project selection.

3.2.2 Qualitative versus quantitative

My research questions are predominantly concerned with the cultural and contextual assumptions and results of contractual incompleteness; under what circumstances may contracts be less complete than possible and what does deliberately incomplete contracts do to a project? Qualitative methods are appropriate to enlighten these questions as it leaves room for more open reflections and answers. Case studies are deemed appropriate as method when the research questions are *how* and *why* (Yin, 1984) as it is in this case.

As this topic is little researched, there is no clear hypothesis to investigate, therefore a survey would not capture new ideas or topics, which is another indication that a qualitative and open-ended approach would be more appropriate. The research has shifted iteratively between theory and case finding, as discussed in Eisenhardt's seminal article on case study research (Eisenhardt, 1989).

In finding recurring topics and trends in the data, I've used techniques from Grounded Theory (Bryman & Bell, 2007). After going through the notes for all the cases, I mapped then out according to the topics that seemed to be most prominent. This has led to the organization of the cross-case analyses (4.2).

3.2.3 Validity in qualitative research

Could an investigation of software development projects from one single consultancy be said to have any external validity, that is to say something valid about any selection beyond the ones investigated in this thesis? Probably not in the same way as with quantitative research, but then again external validity is of contested relevance in qualitative research. A social context is in itself of a not reproducible nature and the element of interpretation involved. What an exploratory multi-case study could bring forth, are topics and propositions for further research.

But how then to appraise the quality of the research? Credibility is one quality measure of qualitative research (Bryman & Bell, 2007). The study has credibility in so far as there is triangulation both with regards to samples, respondents, methods of investigation and also theoretical fields. I have conducted some respondent validation as I've have conducted follow up interviews with several of the respondents. Records of observations and interviews are fairly thick (recorded and semi-transcribed) and all records are kept. Since I work in the field of software development myself, I have tried to keep my own opinion on the matter out of the way and deliberately asked "devils advocate"-questions to the respondents in order to catch any inclinations on part of the respondents to give me the answers they think I would want or that are most valued somehow.

3.3 Selection of projects

I have done multiple case study with four projects with four different clients by the same consultancy. The projects selected have some commonalities and there is also considerable and significant heterogeneity.

One commonality is that they are all consulting projects conducted with participants from the same consultancy firm. The consulting firm was chosen for two reasons. The most obvious reason is that access to projects that was granted me because I work there. The second, and more interesting, reason has to do with the likelihood of finding samples of interest to this thesis. The consultancy in question does not have a specific project methodology that all projects are required to adhere to. It has also been at the forefront of the agile software development trend, and is for those reasons a place where you could expect to find samples of lightweight and even experimental project methodologies.

Another common feature is that all the projects are governed by the same type of contract (T&M) and are of a fairly long duration. These commonalities limit the number of extraneous variables. The fact that they are all projects with external suppliers also assures a certain degree of external control. The contract type and the duration of the projects are also selected because this is where one could expect to find examples of what I'm studying; low levels of specification, planning and estimation.

Some heterogeneity lies in the different kinds of products, sectors, ownership (public, private, large, small), project phase and relation between buyer and supplier.

But more importantly there is significant heterogeneity in the variables relevant to this thesis; the four projects use distinctly different methods when it comes to planning; how when and why they plan, estimate and control the projects. The level of contractual completeness is different at the level of estimates, planning and control, but not at the level of the actual fiscal contract.

3.4 The interviews

I have performed semi-structured interviews with minimum three people in each project. I have chosen to semi-structured interviews as this opens up for new ideas and personal reflections the respondents may have. I have allowed the interviewees to speak freely on the topics of planning, control and predictability first, then used an interview guide to make sure I got some coverage on all the central topics. The interview guide (in Norwegian) is available in appendix A (8.1).

The respondents fall into three categories; Project Managers, developers/team members and customers representatives in the roles of Product Owners. I will in the following refer to the supplier Project Manager as PM and the customer counterpart as PO (Product Owner).

The interviews were conducted in Norwegian. In all there were conducted 15 interviews from this sample, three from each project over a period of four weeks in the autumn of 2013. The exception is project B where I interviewed three developers and an interaction designer. The interactions designer was relevant because he is in many settings the counterpart of the PO in terms of planning and describing the solution. I decided also interview three developers here as the project consists of several different projects that are different with respect to planning and estimating. I also had follow up rounds with all the projects to clarify responses or ask for additional information.

All interviews were recorded and pseudo-transcribed shortly after, the 15 interviews and the four observations resulted in 93 pages of transcribed material.

3.5 The observations

In observing the teams I was looking for how they talk about prioritization, value creation, time limits, scope and size. Is there any correlation between types of controls and how these topics are addressed? I observed specific situations where planning and prioritization is done; explicit planning meetings or more informal discussions about scope, features and value. Where they focused on cost or value in prioritizing what features to develop? Was there a sense of sanctions if certain contingencies occurred?

The observations were ethnographic in nature. My proposition is that cultural and social aspects such as trust, closeness and psychological contracts between the parties impact the degree of contractual completeness (in addition to complexity and volatility). Ethnography is believed to be better at "understanding work organizations as cultural entities" (Bryman & Bell, 2007)

The sampling of situations was based on capturing the situations that could shed light on my research questions. They were used in order to inform and provide prompts for the interviews. Thus I conducted a round of observations before the interviews, collected field notes and photographs of the settings.

The observations are quite short in length. A disadvantage with relative short observations is the possibility of misunderstanding the situations and what they mean. But since I know the (type of) contexts rather well, I'm not concerned with not understanding the setting. What I needed to be wary of on the other hand, is idiosyncrasies and to not take what I saw for granted. What remains a limitation is that the observations were to short to investigate the more complex interactions and processes between the project and the buyer organization.

I adopted an observation style quite close to pure observation; I did not intrude on the situation, did not ask questions or interrupt the proceedings in any way. Neither did I have a role in the context other than that of the observer. It needs to be taken into account that the setting was probably somewhat altered by the fact that I was there observing (as they were all aware of). This may have resulted in them focusing more on the topic they knew to be of my interest. On the other hand, these settings are often conducted with some kind of foreign observer. It is not uncommon that an external party is present at planning sessions, demo sessions, not in terms of research, but as stakeholders that do not have a permanent role in the project.

4 Results

4.1 Within-case analyses

4.1.1 Project A

Core facts

- Sector Logistics
- Project duration 3 years
- Team size 13
- Sourcing From one consultancy
- Location Everybody co-located
- Contract T&M
- Release rate Weekly

This project develops a product to both the B2C and B2B market. The team consists of 10 developers, 1 business analyst, 1 tech lead/project manager and 1 interaction designer in addition to the product owner (Director of Product Development) and the product's CEO. They are all co-located in one open office space. The co-location, continuous collaboration and conversation going on is pointed to as the source of trust in the project. All the interviewees describe the project as being very low on administrative overhead and that this is a very deliberate and valuable aspect of the project.

What is to be developed, the so-called backlog, stem from ideas and requirements from a diverse group of stakeholders; the customer organization, their customers and the development project. Ideas are gathered and categorized on "boards", which are digital task lists with support for prioritization, adding details, estimates, due dates, pictures and additional information (Figure 7).

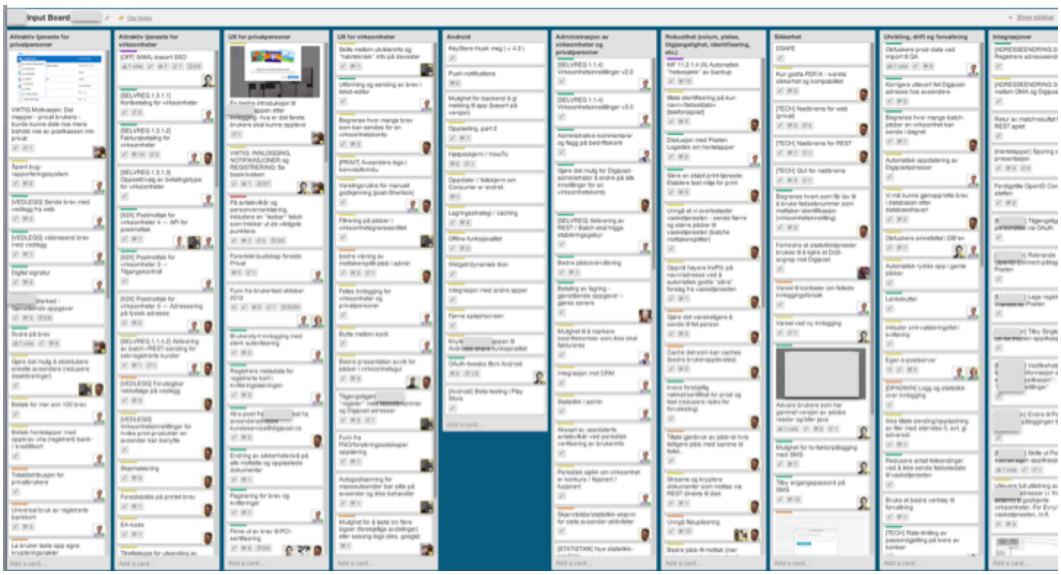

Figure 7 The project "Input board" is where ideas and suggestions for new features and improvements are collected. The different columns represent different target groups, functional modules and/or technical capabilities.

The PO can pull ideas from this idea-board into an operational mode, a Kanban board they call their "Development board" (Figure 8). This board has a workflow structure, features or ideas move from one step to the next; from analyses through to "In production". Tasks move along the board or workflow by a pull principle, inspired by Lean manufacturing.

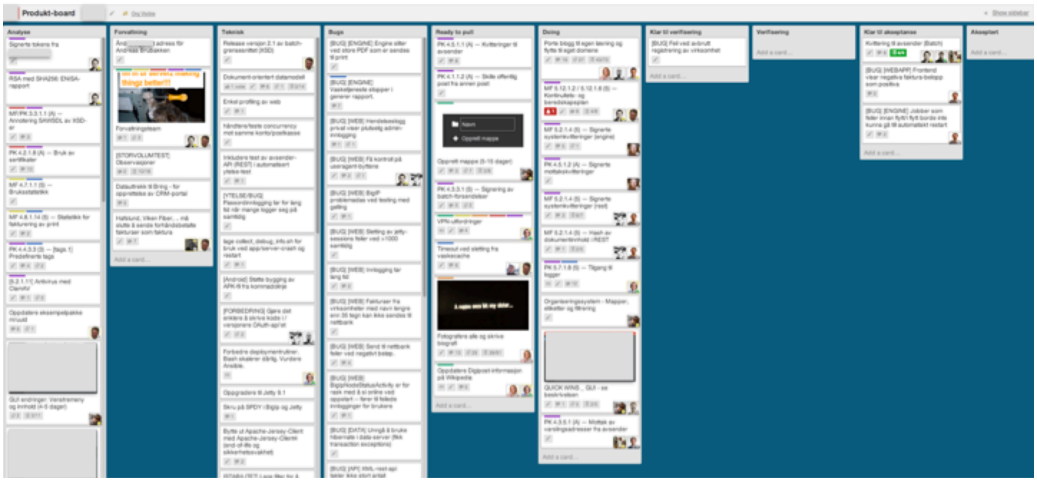

Figure 8 The project's Kanban board. The columns represent the different steps in the workflow.

When an idea is chosen from the "Input board", the team starts to analyze it. The PO and the interaction designer are normally the ones involved in this. This is when the Lead Developer estimates based on a rough breaking down of what work

needs to be done and what stages the feature has to go through. He knows that he is optimistic, so he "pads the estimates". When the analysis step is done and the task has an estimate, the PO can decide to pull it out of the development board if he thinks it shouldn't be developed at this point. That rarely happens though, normally the task starts its journey down the workflow.

A drawback with this pull-based method is mentioned both by the PM and the Lead Developer. Given the queue-oriented methodology, there is little space for the team to get a feeling of the overall mission and vision of the product. They have good visibility in the short term, but neither focus nor practices that address the longer term and the big picture.

In their usual project methodology, the estimates are used for prioritization in order to decide whether to build the feature (as described at that point) or not. It seems only to be relevant as an order-of-magnitude-estimate: "*I need to know whether it's 5 or 50 days work. It demands more of us if it's 50*" [2] PO, project A

Maybe because this is a product for the market and the PO has insight into and/or control over the entire value chain, he sees the development effort estimates as not the most central cost when he considers cost/benefit*:*

> *"We often try to think "what costs are associated with this thing" in a way. Transaction cost, technical cost, increase in staff at the call center. All kinds of costs connected to operations and maintenance. The development itself isn't that interesting, we never look at that"*[3] PO, Project A

Currently however, the process is slightly different with regards to planning, scheduling and estimation. The project has a deadline to deliver certain functionality to an important B2B customer. The deadline is absolute and they have to make sure they are done in time. They still estimate as usual, but they track estimates towards the scheduled deadline. This puts more pressure on the project than usual and they talk about estimates, metrics and efficiency in another way than before. Both the fact that they have (temporarily) changed their process and how that influences how they work are interesting discoveries. Whereas they under "normal" conditions speak of changing priorities as a good thing, the PO expressed

---

[2] Originally: *Vite om det er 5 eller 50 dagsverk. Det krever mer av oss hvis det er 50 dagsverk*" PO, Project A.

[3] Originally: *"Vi gjør ofte en sånn "hvilke kostnader er det knytta til dette her" på noe vis. Transaksjonskort, tekniske kost, økt bemanning på kundesenter. Alle mulig type kostnader knytta til Drift og forvaltning av det. Selve utviklingen er ikke så interessant, det ser vi aldri på"* PO, Project A

the opposite stance when it comes to the current deadline: *"we've tried not to re-prioritize now"*[4]. They also calculate that they now spend 10% more than usual on administrative work because of this deadline.

All interviewees and also the planning meeting I observed discussed scoping down features. Scoping down was considered valuable to get something to market as soon as possible:

> *"An important point is that we try to reduce the scope of the tasks, that's part of our agile methodology. As small as possible, a Minimal Viable Product kind of thinking."*[5] PO, Project A

Actual development effort is not ordinarily measured against the estimates. If there are significant delays this is picked up during the daily status meetings as a task lingers in "Work in progress" If a task seems to take longer than anticipated, the team discusses if the scope could be reduced or whether it could be divided somehow. All respondent also mention the culture in the project for asking for help if one is stuck on something.

The PO and the PM both have reflections on what measuring by estimates can do, the PO states, that accuracy in estimates is not the same as a successful product:

> *"That's the hard part with estimates. The negative part of estimating. If we hadn't used estimates at all, we would have developed, been done and been happy with the result. The feature is super good, solves what it's supposed to solve. The reason there is discontentment now is that there is this theoretical construct around that it's supposed to take 5 days, it takes 25, and that leaved the impression that it's somehow a failure. The planning is a failure, but the execution is a success. The product is good. It's important to keep those two (concepts) apart."*[6] PO, Project A

---

[4] Originally: *"Vi har prøvd å ikke omprioritere nå"* PO, Project A

[5] Originally: *"Viktig moment er at vi forsøker å scope oppgaver ned, det er en del av smidig metodikken vår. Så lite som mulig. Minimal viable product tankegang."* PO, Project A

[6] *Originally*: *"Det er det som er vanskelig med estimater. Det negative med estimering. Hadde vi ikke brukt estimater i det hele tatt, så hadde vi utvikla, blitt ferdig og vært dritfornøyd med resultatet. Funksjonen er superbra, løser det den skal. Grunnen til at det nå er misnøye nå er at man har lagd teoretisk konstruksjon rundt at det skal ta 5 dager, tar 25, inntrykk av mislykka. Planleggingen er mislykka, men gjennomføringa er vellykka. produktet er bra. Viktig å skille de to fra hverandre."* PO, Project A

4.1.2 Project B

Core facts

- Sector Public administration
- Project duration 3 years
- Team size 12
- Sourcing From one consultancy
- Location Team co-located, PO in different city
- Contract First incentivized Fixed price, now T&M
- Release rate Weekly

This engagement between buyer and supplier consists of several projects. At the point of my research, there were two projects running in parallel in addition to the maintenance of previous project deliverables. One of the projects estimate, the other does not. They all have the same PO in addition to domain specific personnel that are allocated to the projects for requirements specifications, follow-up and testing. Customer personnel and PO are not co-localized with the team.

At the start of the relationship, the customer deliberately chose an incentivized fixed price contract in order to incentivize the supplier and also to build trust. As the project was successful and trust was built, the contract has been a T&M contract. This a very concrete example of the TCE term Fundamental Transformation.

Now when new projects are initiated, the customer expresses quite high-level needs and expects a high level response on how much time (and money) development will take. The interaction designer and the PM are responsible for creating the overall concept and the UI designs. The solution and designs are presented to the entire team; this has been the basis for estimates made by the team (mostly the entire team). This constitutes the designs- and specifications phase.

The estimates are given in Story points and are visible for the developers as it is documented on the tasks on the digital task board. Developers also report hours spent on each task/feature. In this way, the PM can map story points to hours and hard currency in retrospect. The PM uses this metric to give some prediction on how much time delivering a number of story points will take.

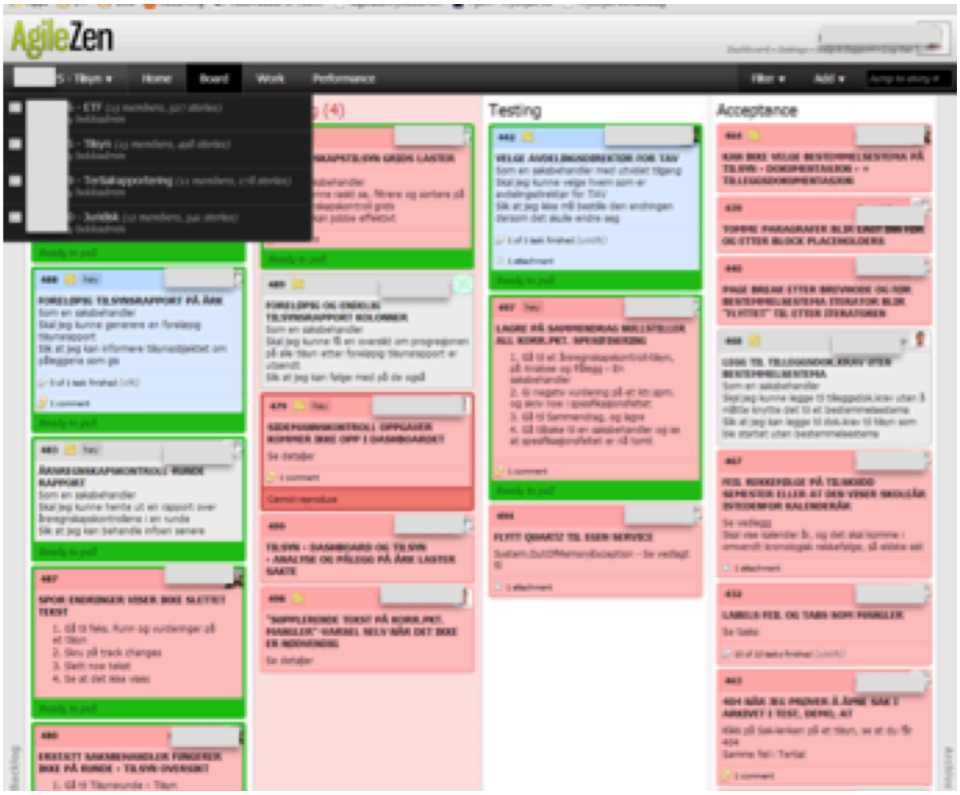

Figure 9 Project B Kanban board. The different colors represent the different concurrent projects in addition to maintenance of previous deliverables. The columns represent the steps in the workflow.

This is the only project of the ones I've researched where estimates are actually tracked systematically against actual development hours. This may have to do with the PM's propensity for metrics: *"I enjoy learning about projects by way of the numbers"*[7]. The metrics are not used directly in performance reviews or any other way, but it seems the developers are quite aware of being tracked in this way.

In the engagement's latest projects, they have skipped estimating features or tasks altogether. The rationale for skipping estimates was that they were not really necessary and the project felt it could spend more time on productive activities instead. The PM talks about it as an experiment, he is curious to see if the project can still be controlled sufficiently. The customer is still given an overall cost estimate based on historic data and experience, but there is no estimation done by developers. The estimation meetings are thus skipped. The interaction designer details the features and tasks. The tasks are documented on the digital task board (Figure 9), where a developer picks it up when she/he is available for a new task.

[7] Originally: *"Jeg synes det er gøy å lære om prosjekter tallveien*" PM, Project B.

As with project A, the process with regards to estimation and scheduling as thus changed (during or between projects) to fit the current context better.

There is an interesting and surprising finding here. Even with the non-estimating project, there is still tracking going on. The difference is merely that the PM assumes that in the long run, all user stories are more or less the same size. Since the tracking method is the same (story (points)/hours), this has some interesting consequences. For example if you split a story in two - which can be a valuable mechanism for getting something valuable into production sooner - this is tracked as an increase in scope, a scope creep. And since there is an (high-level) estimate given the customer at the onset of the project, this is a problem. One team member comments on this as he describes how he thinks about having to split or make new stories (as new needs emerge):

> *"Earlier it was easy to split stories (with estimates and story points). Now I increase scope by splitting, and the PM tells me scope has increased. So I describe larger stories in order not to increase scope. It's a side effect."*[8] Team member, Project B

The removal of estimates also affect the PM's ability to make earned value analyses and predict time of completion and remaining work, ETC (Estimate To Complete). This presupposes a scope that is fixed, or at least a scope that it is possible to track in terms of size. The PM knows this is a fault of the tracking, and not of the practice in itself:

"We lose something tracking wise. But that's tracking wise, we do get the same deliverables."[9] PM, Project B

The developers express that they are happy about not having to estimate – and not being tracked by estimates in the project:

> *"It's good because we don't have the discomfort of estimating as an obligation. (…) you promise something personally that you cannot guarantee. I don't have all the variables for*

[8] Originally: *"Før var det lett å splitte opp Stories (med estimat, poeng). Nå øker jeg scope ved å splitte opp, da kommer prosjektleder og sier at vi øker scopet. Dermed så spekker jeg store stories for ikke å øke scopet. En bivirkning, da."* Team member, Project B

[9] Originally: *"Mister noe trackingwise. Men det er jo trackingwise, man får jo levert det samme sikkert."* PM, Project B

promising that. My experience shows that there comes new variables along the way"[10] Developer, Project B

However, with this new practice of not estimating, several of the developers express missing the big picture. Since they no longer have the planning meetings and estimating sessions, they never get to take a step back and look at the overall product, vision and goal:

"It feels like a production line and nothing is ever finished. There are no milestones that I know of because the Customer is removed, the specification is removed. I sit in the middle and just work. But I should maybe be careful of what I wish for."[11] Developer, Project B

As with project A, the PO states an aversion to administrative overhead, explicitly wanting no non-productive roles in the project.

The steering committee meets relatively seldom, earlier on they met four times a year, now not more often than twice a year. There has never been a point where they were called upon to make a critical decision, but are simply kept informed.

For the PO, getting working software out to the users as soon as possible is highly valued. Not only to reap benefits as soon as possible, but also in order for the users to better understand what they need and what further software could help them. The idea is to get a basic version of something that could work out there, and then start gathering the "real" requirements and needs:

"The needs are not what you think they are, but what appears when you start using it"[12] PO, Project B

Delivered software then becomes part of planning or a tool for planning.

---

[10] Originally: *"Bra fordi vi slipper det som er den ubehagelig som er estimering, som en forpliktelse. (…) du lover noe personlig som du ikke kan garantere. Jeg har ikke alle parameterne for å love det. Min erfaring viser at det kommer nye parameter langs veien*" Developer, Project B

[11] Originally: *"Føles samlebånd og leverer aldri noe ferdig. Ikke milepæler som jeg har noe forhold til fordi kunden er abstrahert bort, spekken er abstrahert bort. Sitter i midten og bare jobber. Men kanskje skal jeg være forsiktig med hva jeg ber om."* Developer, Project B

[12] Originally: *"Behova er ikke det som man tror man har, men det som viser seg når man tar det i bruk"* PO, Project B

4.1.3 Project C

Core facts

- Sector Transportation
- Project duration 1,5 years
- Team size 9
- Sourcing Multi- and internal sourcing
- Location Co-located
- Contract T&M
- Release rate Weekly

The project developed an Intranet and it was considered time critical to replace the existing one, which was seen to be very poor. At the point of this study the project had recently launched version 1.0 of the intranet and were still in project mode whilst developing enhancements to the existing modules.

Before the project got the final go-a-head, there were several preliminary design- and concept phases in order to get funding and to reach a consensus in the organization of what the new Intranet was going to be – both conceptually and technically.

Funding was given on the basis of high-level analogy estimates; comparisons with previous projects, adding buffer for known risks and uncertainties etc. There was in other words no software development effort estimates made, which is the definition of estimates in use in this thesis, ref. 2.2.2. Funding was considered (by both PM and PO) to be generous/ample and scope of work was described on a high level. As the PM states: *"But the goal is to give room to do things afterwards."*[13]

Technology uncertainty is considered rather large and the project has been trying to choose solutions that work with the chosen software platform instead of keeping strictly to an ideal concept. The technology platform is large and complex and there were several new and unknown technology components. There is also considerable uncertainty and external dependencies connected to content production and content migration.

The development team consists of 5-7 developers from different consultancies and internal resources. The administrative overhead in the project is low, all the team members has to do to enable tracking, is to note the date on the index cards when they move them along the workflow (see Figure 10).

[13] Originally: *"Men målet er å gi et rom for å gjøre ting etterpå."* PM, Project C

The entire team was co-located with the PO during the main project, but after the release of version 1.0 the PO is less available and this is considered a problem. Both the developer and the PM express the continuous co-location with the PO as crucial. He is accustomed to this kind of agile process and knows the domain and the organization and user needs very well.

Features are only broken down into workable and prioritizable items shortly before they are developed, a rolling wave type of planning. There is no estimation in this process, but scope and complexity is talked about in order to find an optimal solution; *"This way to solve it is quite complex, can we look at an easier way to achieve the same?"*[14] This way of discussing scope, cost and duration is different from giving an estimate and getting a "go" or "no go" in response. It is about exploring different options and different ways to solve – or interpret – the problem. When planning is done this way, one could be said to utilize and take advantage of the ambiguity that is inherently there when professionals from different background and different roles work together on a domain with a considerable uncertainty and complexity. Cost/benefit is talked about on an "off the bat"-kind of way and neither documented nor tracked.

A white board (Figure 10) with index cards visualizes the workflow and is used to keep track of work in progress, much like the digital boards of projects A and B.

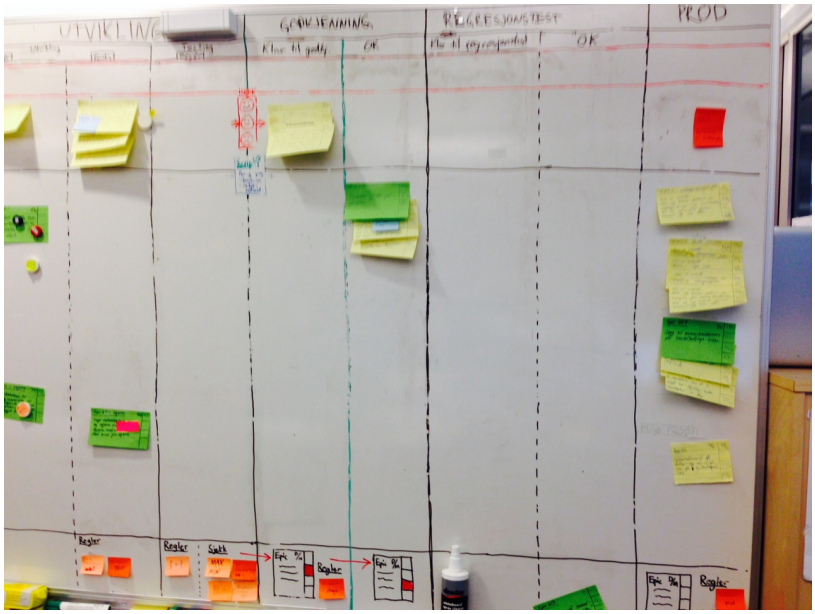

Figure 10 Kanban board in Project C. The columns represent the stages in the workflow. At the bottom of each column there are rules pertaining to each step.

[14] This is not a presise quote, but gathered from the observation of the team at work in a planning session.

Even though development cost is not directly measured against any estimates, the PM translates progress into the organizations preferred metrics; an earned value analysis. Instead of measuring estimates to actual development effort, the PM tracks cycle time (output rate) and lead-time (the time it takes a task from it's being analyzed until it's in production). Over time, this gives him a good dataset to predict future performance. The project releases to a demo (and after the beta-release production) environment every week, so the feedback loop is short.

There was no fixed release dates at the outset of the project, but dates were set as the project got an idea of when they could have a beta version and a first version ready. The dates were set based on a prediction model of progress and also by choosing some "mental threshold dates" as the PM calls it; before Easter, by summer, before Christmas. All the while the scope was allowed to increase; ideas came in from the organization and the produced software triggered ideas;

> *"When we released into the production environment, it triggered thoughts and new tasks were added"*[15] PM, Project C.

This was allowed to happen and the backlog thus grew. As the set release dates neared, the PM and the PO together "shaved" the backlog or cut scope, prioritizing what had to be part of the release (Figure 11):

> *"At an early stage people can't understand what doesn't need to be part of a first version"*[16] PM, Project C.

[15] Originally: *"Når vi lanserte ut i produksjonsmiljøet, så trigget det tanker, så kom det nye oppgaver"* PM, Project C

[16] Originally: *"Folk klarer ikke i en tidlig fase å se hva som ikke trenger å være med til en førsteversjon."* PM, Project C

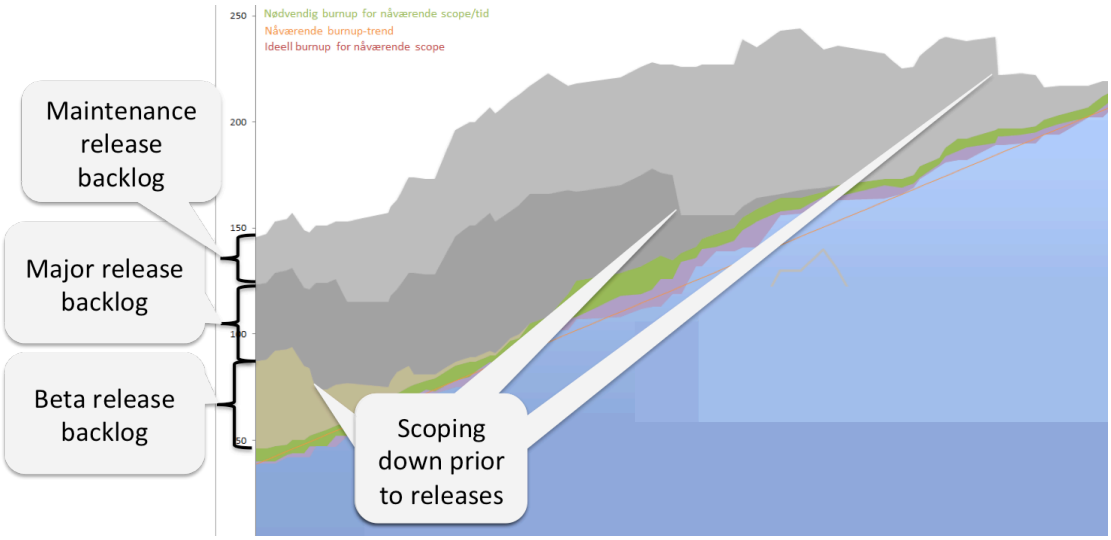

Figure 11 The drops in the beige and dark grey areas indicate the cutting of scope before each release.

There are two interesting points here. First, the customer and the project did not experience the cutting of scope as problematic or painful. They instead saw the value of being allowed to change priorities and discover new needs as they gained knowledge of the product and the solution emerging. Secondly understanding what features were *not* needed after all was as important and recurring as discovering new needs. The ability to change priorities and make room for new features or needs was considered more valuable then getting "everything".

The scheduling process here is similar to the Microsoft "Sync and stabilize" process described in chapter 2.7.2 with one important difference; the scope was allowed to grow and change throughout the project.

Getting an early version out to real users was considered paramount:

> *"… get a first version as soon as possible. This enabled a beta version half a year before the real launch."*[17] PO, Project C

[17] Originally: *"…få opp en første versjon raskest mulig. Muliggjorde beta-versjon halvt år før vi lanserte."* PO, Project C

4.1.4 Project D

Core facts

- Sector Finance
- Project duration 3,5 years
- Team size 10
- Sourcing Two consultancies and internal resources
- Location Co-located
- Contract T&M
- Release rate Monthly

The project delivers internal software used by caseworkers all over the country. It has been in production for several years and is being expanded and improved by the current project. The project spends 20% of its resources on maintenance and smaller enhancements of existing functionality; the rest is development of new modules and features. The current project has several integration points to other systems and is depending on deliverables from other internal and external projects. This is increasing the complexity and uncertainty of the project, especially schedule volatility and uncertainty due to external dependencies and deliverables.

The project consists of a mix of external consultants and internal technical and business resources. This is official policy in the company, probably in order to transfer and keep knowledge within the company. The access to and participation from end-users and business side stakeholders is expressed as a positive side of working in this project.

The project falls under the company's portfolio management and it's governance, this influences financing and processes. The project gets funding and operates for 9 months long phases, and is presently in phase number five. Funding is based on a business case. The benefit side is a calculation of the savings or increases in efficiency gained by each feature, the cost side of analogy estimates on medium level granularity. This planning and project charter process starts about 6 months before the actual phase starts and is described as rather high-level and complex as it involves many stakeholders.

The estimates given in the charter phase are what the project in measured against by the steering committee, which meets every two weeks. There is also another stakeholder group that helps the project sort out its priorities; the prioritization forum. They help the project make sure that the right features are prioritized for each release.

Figure 12 Kanban board of Project D. As some tasks increase in complexity or detail is added, there are post-its added to some of the white task labels. (I did not get an explanation for the lay appended to the board.)

The project releases new features and improvements once a month. When a feature is put in queue for the next couple of releases the functional team starts breaking it down into more detail. They describe the feature in text and UI designs. Then the architect and some of the developers (a relevant mix of domain knowledge and seniority) estimate. As they estimate, they do not have access to the initial estimates made in the charter phase. The PO uses the estimates to deicide on whether or not to prioritize developing the feature now, changing it or discarding it for the moment;

> *"It doesn't happen that rarely that we take tasks out, postpone them…It's when we deliver functionality that the change requests arrive"*[18] PO, Project D

This quote also shows another interesting, and recurring, topic; delivering software leads to the need for new features. Change and change request is though talked about in a slightly more problematic way in this project. The interviewees express that larger changes, change requests or scope creep can be problematic to deal with. This is also given as a reason for having 9-month phases; it's short enough so that larger changes can be put on hold till the next charter process and phase.

[18] Originally: *"Det hender ikke rent sjelden at man tar ut noen oppgaver, utsetter de. … Det er at når vi leverer funksjonalitet, det er da endringsønskene kommer"* PO, Project D

> *It seemed like a good duration of a phase, because too many things don't turn up that makes us have to change very much. But some things do turn up, but they have been put off until the next phase.*[19] PM, Project D

Smaller changes can be implemented as part of the 20% maintenance slot. Adding large changes during a phase requires updates to the business case and thus requiring decision by the steering committee. This is described as an uncomfortable process.

Development time is not tracked explicitly, but scope is discussed during the daily stand up meetings. If tasks are more complex than anticipated, they may be split into several smaller tasks and re-prioritized. Increase in complexity is mostly accepted and taken note of.

[19] Originally: *"Det har virka som ganske fin lengde på fasen, for da dukker ikke opp for mange ting som har gjort at vi har måttet endre veldig mye. Men det har dukket opp en del ting, men det har kunnet latt seg vente til neste fase."* PM, Project D

4.2 Cross case analyses

The projects at a glance:

	Project A	Project B	Project C	Project D
Sector	Logistics	Government	Transportation	Finance
Contract	T&M	T&M	T&M	T&M
Sourcing	One consultancy	One consultancy	Multi-sourcing	Multi-sourcing
Estimates	Some	Sometimes	None	Twice
Estimate use	Prioritization Scheduling	Bidding Prioritization Control	None	Business case Prioritization
Estimates timing	4 weeks before	Start and during	-	6 months before 4-8 weeks before
Estimate currency	Days	Story points None	-	Story points
Who estimates	Lead developer	Team or part of team	-	Part of team
Release frequency	Weekly	Bi-weekly	Weekly	Monthly

Based on the data gathered, there are some topics or focus points that stand out. Inspired by grounded theory (ref. 3.2.2), I've chosen to organize the cross-case analyses by the topics that stood out.

4.2.1 Up-front planning and estimating

Contractual incompleteness is linked to the activities and phases prior to the start of the actual project. The output of a design- and specifications phase is what corresponds to *ex ante* specifications or detail in contract theory. How complete are the contracts or project charter in the four projects I have researched? And how does the up-front specified plans color the project?

The four projects are heterogeneous with regards to up front planning and estimating, but it isn't simply a question of estimation or not. Only one of the projects has done no estimating at all, whereas the other three projects have

varying degrees of estimates. It also varies why they plan and estimate as they do. As discussed in 2.2.2 there are five categories of use of, or reason for, estimating; bidding, business case, project planning, prioritization and control. Whereas project A usually estimates in order to prioritize close to execution, it now also uses estimates to schedule (aka project planning) for a specific deadline. Project B uses estimates for several reasons; bidding, prioritization and control, and project D makes two different sets of estimates for two different purposes; business case and prioritization. But since the project reports according to the business case, there is also a non-explicit control element here.

At what point in the process they plan and estimate also varies a great deal and divide the projects in two camps; whereas project A and C has planning, specification and estimation very close to execution, both projects B and D have a specifications and estimation phase prior to starting the project or phase.

There is even significant difference in how they plan and how plan-driven they are; project A and B have low level of ex ante detail and little ex ante process. Project C had a big ex ante process, but not very much detail. Project D has both process and medium level detail ex ante. According to research on contractual incompleteness in domains with uncertainty and complexity (ref chapter 2.3.2), one would expect that ex post changes are more problematic when ex ante specification is high. And this seems to be the case with project D, it is the only project where change is talked about in terms of being a challenge. In order to accommodate larger change requests, they need to put them in the backlog for a later phase. I will come back to this topic in chapter 4.2.2.

Project C had, as mentioned, a long planning phase, but the charter was not very detailed and thus not considered to be very controlling of what the projects could deliver. The charter (aka the project contract) was in this sense quite incomplete. Especially technology uncertainty was considered to be high. Both the technological uncertainty and changes in market needs (user and stakeholder feedback and ideas) could be taken into consideration during the project.

A second question is also inspired by the literature on contractual incompleteness: is there a correlation between the level of ex ante detail, or up front planning in our context, and the make-or-buy decision? What make-or-buy decisions did the corporations in my sample make? Is it strictly one or the other, or is it somewhere in between? In my sample, I have projects conducted with contracted resources or entire teams, which means all the customers have chosen to contract their project to a certain extent. But even with this apparent similarity, the four projects have different sourcing strategies. Two of the projects, C and D, have an explicit rule of having internal resources in the project. In project D they have some roles that the organization require to be internal, such as Solution Architect and PO. They also require there to be at least one developer from their own pool. Project C had a

position of Technical Lead tailored and recruited to fit the project's and product's need. Both of these choices can be understood in the light of information stickiness. It can be seen as a way to ensure that the developed knowledge remains in the organization (ref chapter 2.6.). Whether the transference of knowledge and information is actually successful will be influenced by the organization's ability to capture, assimilate and utilize new knowledge, its absorptive capacity.

These findings point towards a hybrid integration model; you hire in the market to solve the problem, but you also develop your own organizations skills. Instead of contracting the entire system, you hire competency and experience. This can be seen as a partial vertical integration.

4.2.2 Change and uncertainty

There seem to be a large difference in how the projects deal with change during the project life cycle. In projects A, B and C change is mostly talked about as something positive, it represents learning and something the process is geared towards. In project D on the other hand, change is dealt with in the specifications phase. 9-month phases are described as a good length of time because then too much change does not occur, and those that occur can wait for the next phase. Implicitly change is something that is difficult to accommodate during a phase.

Change does not only occur in the product features, it also occur in the project's own processes. As mentioned in 4.1 both project A and B has recently changed how they use estimates. Project A has had to rely on estimates not only for prioritization, but also for scheduling and hitting a hard deadline. The PM talks of this as a *"state of emergency"* and continues:

"I thought we could work in the same way now too, but it is a bit different with these constraints, that everything must be done"[20] PM, Project A

The projects attitude towards change and changes in priorities is highly affected by this temporary regime; they try not to change. This is an important cue to what rigid estimates for the sake of scheduling and control does to a project.

Quite the opposite is the fact with Project B, they have stopped using estimates altogether in their latest project – simply because there was no need for it and it was considered a waste of time.

For two of the projects that have low level of estimation and planning activities (A and B), team members mention missing the bigger picture. A couple of developers

[20] Originally: *"Har trodd vi kan jobbe på samme måte nå også, men det blir litt annerledes med de rammene, at alt må være ferdig."* PM, Project A

also mentions feeling almost disenfranchised, not connected to the customer and the value that is to be created:

> *"I wish I had more contact with the customer, I don't really know who I'm making this for anymore"*[21] Developer, Project B.

By eliminating more planning processes, they have lost sight of why and for whom they are making the product. In an effort to reduce waste, some valuable practices might have been accidentally left behind. PM of project A and B are aware of this and have different strategies to remedy it.

Another finding that reoccurs in the least planning projects, is the value of scoping down, reducing each feature to its most basic core. This is considered valuable for a number of reasons. One key point is to start capturing value of the software sooner rather then later. In the case of replacing an existing system or existing manual processes, even a rudimentary solution is considered better than status quo.

Another aspect that is mentioned in all the projects is the ability to learn from an early, maybe even premature product release. Launching of "beta" versions is based on this idea. Getting only a small part of a solution out to real users can give important input to the further development process. And this input is obviously easier to take into consideration if the project is not constrained by detailed up-front requirements.

A third value from early releases was more surprising to me. A PO talked about getting software out in order for the organization to understand what they need.

> *"All experience tells us (also in this project as in every project) that when you start using the solution, the expressed needs aren't as important anymore. When they start using the solution, they see other needs that are a lot more important"*[22] PO, Project B

In a sense, working software becomes a catalyst. In a classical project model the requirements specifications are the objects of planning, but here the software itself becomes the object of planning. I will discuss this further in chapter 5.3.5.

How do the projects deal with technology uncertainty? Projects B and C are explicit about this. They have chosen a technology standard or platform, and are aware that this has both advantages and disadvantages. They both express having

---

[21] Originally: *"Jeg skulle gjerne hatt mer kontakt med kunden, jeg vet ikke lenger helt hvem jeg lager dette for."* Developer, Project B.

[22] Originally: *"All erfaring tilsier (også dette prosjektet som i alle prosjekter) når man begynner å bruke løsninga, så er ikke uttrykt behov så aktuelt lenger. Når man begynner å bruke løsninga, ser de andre behov som er mye viktige."* PO, Project B

made a deliberate decision to work with the technology and implement features according to "best practice". If a feature is difficult to implement exactly the way one had envisioned, but the technology standard has another way of solving the same problem, they have chosen to stay with the standard solution. This leads to changes in the concepts and features from what was "optimal", but decreases the cost of both development and maintenance.

4.2.3 Organizing and interaction

As already mentioned, it is not clear cut whether the four projects are really cases of "making" or "buying" on part of the customer. There is a varying degree of vertical integration or at least "body shop" consultancy hire under the complete control of the client.

The presence of the customer throughout the projects seems to be stable and mentioned as a success factor. Overall the customer participation is high, and if not sufficiently present, a major impediment. The customer's domain knowledge and input is crucial to the making of the software, and co-location is recurring as the best way to accommodate this. And in fact all project except Project B have the PO in site. This seems to be balanced by the fact that the trust and understanding between the PO and the PM is strong.

All four projects report of having a high level of well-being and job satisfaction. Two of the PO's mention the team's motivation and job satisfaction as something they greatly value and respect.

There are different degrees of formality and ceremony in the projects, but overall the level of formal processes and procedures are low. The Kanban-style pull based workflow seems to accommodate this very well, and is consistently mentioned by the team members as a successful practice. The projects are themselves not affected by the corporate project or portfolio governance. In project C there is a rather rigid structure of project governance in the organization, but the PM seem to be the buffer for this and the project is hardly affected, if even aware of, it.

Experience of management and team will also be likely to play a role here. It's probably easier to relinquish some control and accept open-endedness if you are experienced and know the field well. Three of the project managers in my sample are rather experienced; their experience ranges from almost 20 years to 8 years. The least experienced PM has three years experience as a PM (but has 5 years experience as a software developer, so he knows the field and type of work well). His project is the one with the most up-front planning and estimating, but that seem to rely on other factors; corporate culture and the organization's portfolio management regime as described in 4.1.4. As for the customer side, all the POs are quite experienced in their roles and in their organizations. This is a correlation that would be interesting to investigate by way of quantitative analyses.

All the PM's and PO's were asked what they expect of the other and what they think the other expect of them. My findings were to a large extent in line with Koh et al. (2004) findings; the supplier expects the customer PO to provide clear specifications and have (and share) available knowledge about the business and the domain. Another recurring expectation of the customer is their ability to make decisions and to prioritize. The customers expect the supplier PM to have control of resources and staffing and that the team is working and thriving. In general the supplier's PM think they are expected to have a larger degree of control than the customer's PO actually expresses. The PO's expectations are expressed more "softly"; to have an understanding of the domain and the technology, to handle the project administration and guide the team. I too make the same finding as Koh et al.: there is no mention of actual deliverables in the expectations from neither the PO's nor the PM's. My interpretation of this is that as mature technology managers, they know that deliverables depend on so many variables and so many people as to be able to expect it of one person or relation.

4.2.4 Performance and project success

There is no common way, or maybe no way at all, to measure and compare the financial performance of these four projects. For one, many of them don't really track performance by financial measures at all, and if they do, the measures are not consistent over time or not available at the time of my (non-longitudinal) study. This will be a common finding in agile projects, they tend to be deliberately low on overhead and tracking. That is not to say that the projects do not think and evaluate themselves in terms of success or failure.

For project A, success is measured in terms of customer acquisition and product adoption and the software development is only a part of the contribution here. For project B success is talked about as passing audits from the Office of the Auditor General and also making the bureaucracy more efficient and less hampered with defects. The effectiveness is not measured directly; it is only assumed that it will easily have been a positive business case. For project C it was the act of replacing the existing solution as early as possible that was the most important success metric. There were also metrics of user adoption and user/employee satisfaction that were still being analyzed at the point of my research.

Project D is the only project that has a clear quantitative business case from which to measure project success. However, according to the PM, this was not straightforward. The organization at large and the different line managers can choose when to reap the benefits the project enables. This means that the benefits are not systematically measured at any one point in time after the software is delivered, but effects can be "delayed" according to how the organization starts actually using the software. When asked how they would know if the project was a success or not, the PO states that:

"We must have done something right since we are in phase five. The business side sees us as the future for them"[23] PO, Project D.

From the supplier's point of view the measures of success are not that different. The best measure of project success is to be able to help the customers achieve its goals and become a trusted advisor and partner in the long run. This ensures trade, builds further trust and thus keeps transaction costs low.

Since employee retention and satisfaction is important for the supplier in question, the team members job satisfaction is also an important success metric.

The supplier organization is not a very financial-metric-driven organization, it has low administrative overhead and doesn't measure project profitability in any rigorous way. One of the reasons for this is that having those numbers available would condition, or prime in the vocabulary of Kahneman, the choice of projects too much. There are other considerations is given more weight then the isolated profitability of the project.

[23] Originally: *"Vi har gjort noe riktig når vi er i fase fem. Da har forretningen ønsket at vi skal… at vi er fremtiden for dem."* PO, Project D.

5 Discussion

5.1 Introduction

I will answer my three research questions in the light of both the literature and the empirical data. The first question relates to what circumstances or contexts contractual incompleteness could be a viable and even valuable practice. Removing up-front specifications and estimates is a radical proposition and it's clear that it's not a feasible option in every setting – or maybe even in most settings. Does my research and the theoretical perspectives explored shed some light on what those settings or circumstances may be?

The second question follows logically; what does removing up-front specifications and estimates do to a project? How does it affect processes, planning, attitudes and products?

The third question has a normative aspect as well; in the absence of traditional planning and controlling mechanisms, what processes and practices needs to be put in place?

5.2 Under what circumstances can contracts be deliberately incomplete?

5.2.1 Trust and reputation

Transaction cost theory tells us to look at the level of internal uncertainty and fear of opportunism when understanding the level of contractual incompleteness. Trust is an aspect that shows up and seem to be built in all four projects at the point of my research. They have all been going on for some time; the buyer and the supplier have had time to build a trusting relationship. But how did they get there?

The clearest answer is to be found in project B. When the engagement started, the buyer and the supplier did not know each other. The buyer needed to ensure that the supplier was trustworthy and able to deliver. In the first project therefor, the buyer used an incentivized fixed price contract to make sure: *"that the supplier is able to do a reasonable job, not abuse the trust, is sober, honest, all those kinds of things"*[24]. It is then reasonable to assume that the internal uncertainty, fear of opportunism was in play here. After this first project, they changed to a T&M contract and higher scope granularity, the Fundamental Transformation had taken place.

Another factor reinforcing trust and reducing the fear of opportunism is the closeness, the co-location and continuous collaboration between the customer and the supplier. Being part of the development process also reduces the problem of software's inherent experience attributes.

Another insurance against opportunism seems also to be at play here; reputation and the expectation of a long-term partnership. Norway is a small market and word travels fast. The consultancy in my sample has a longstanding good reputation for having highly skilled personnel and maintaining long-term partnerships with their clients. This may work as an insurance against opportunism from the buyer's point of view. In accounting one would refer to this as the value of goodwill or in marketing as reputation as a tradable asset.

Is this kind of trust between supplier and vendor more likely to play a part in contracting in societies with high societal trust? Norway is a society with a high level of trust and it would be interesting for further research to see if practices with low levels of up-front planning are tested in societies with lower societal trust.

5.2.2 Governance and transaction costs

Is there any explanation to be found in the organizational factors; how the projects are governed and how they relate to transaction costs?

---

[24] Originally: *"at leverandøren er i stand til å gjøre en fornuftig jobb, ikke misbruker tilliten, edruelig, ærlig, alle de knaggene der"*, PO, Project B

The projects that are closest to vertical integration are the two who have a multi-sourcing model and also have internal (as in employed) team members in developer roles, projects C and D. They have another thing in common: they have the largest specifications phases. In the case of project C, the specifications phase and the actual project were largely decoupled, but as far as the organization is concerned, there was a long design- and specifications phase. This might be the opposite of what TCE describes; here the projects with the most ex ante detail are also the ones who are closer to vertical integration. My data here is obviously neither ample nor strong enough to falsify this fundamental insight from TCE, but it's still an interesting observation. Maybe the make-or-buy dichotomy is too simplistic and that there are aspects of the organization such as dynamic capabilities, absorptive capacity and organizational learning that come in to play here. The design of my research does not allow me to get any real input on those aspects, but it would be interesting to investigate contractual incompleteness in the light of the buyer's capacity for organizational learning.

Another possible precondition for contractual incompleteness may be the attitudes towards transaction costs and non-productive tasks in projects. In common for the projects that use lighter planning methodologies (project A, B and C), is their distaste for non-productive tasks and time. Removing what traditionally was a lengthy design- and specifications phases of software projects will in itself be a considerable reduction in non-productive time. The Project owners of both project A and B are very explicit about this, they repeat it several times and it seems to be core to how they make decisions about process:

"*I strongly dislike non productive time*"[25] PO, Project A

"I'm allergic to overhead, I don't believe in the (purely) administrative Project Manager."[26] PO, Project B

This may be the result of personal preferences of these stakeholders, but it may also be illustrative of the corporate culture they belong to. An aversion to administrative overhead is also a feature of the corporate culture of the supplier in my sample. It may not be a prerequisite for having more incomplete contracts, but it may very well be a better fit with practitioners and organizations that are of the abovementioned mindset.

[25] Originally: *"Jeg misliker sterkt ikke-produktiv tid"* PO, Project A

[26] Originally: *"Jeg er allergisk mot overhead. Jeg har ikke noe tro på den administrative prosjektledern"* PO, Project B.

5.2.3 Uncertainty, volatility and ambiguity

Another topic both the data and the theory on contractual incompleteness suggests as a reason for, or enabler of, incomplete contracts, is external uncertainty. Considerable external uncertainty; both technological and market uncertainty, is indeed mentioned in all the projects. Both the customer and the supplier side also name complexity and volatility as motives for having such planning-light processes: *"Priorities change all the time."*[27] This is in line with the theory on contractual incompleteness and uncertainty explored in chapter 2.3.2. A high degree of complexity and volatility indicates higher contractual incompleteness (see Figure 13).

In project D, the project that has the most up-front plans and estimates, changes in scope is somewhat harder to deal with. They have a high level of uncertainty and volatility, but their contracts are more complete then in the other three projects.

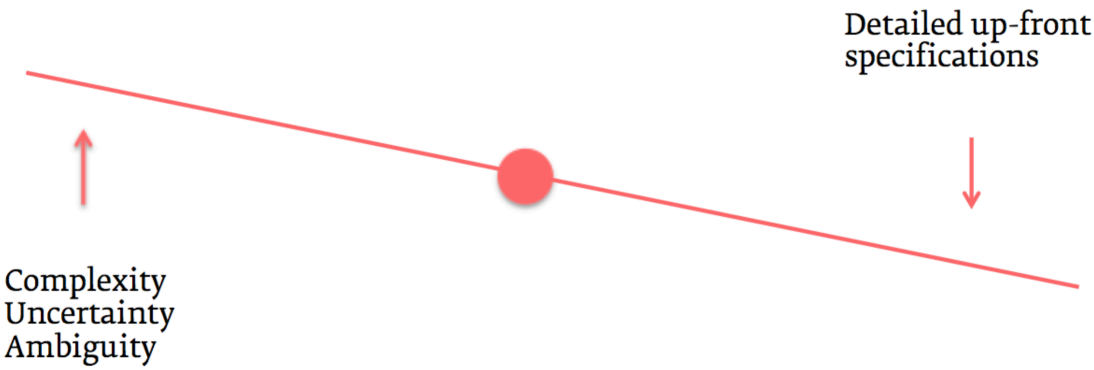

Figure 13 As complexity, uncertainty and creativity increases the level of contractual completeness decreases

There is little explicit mention of creativity in my data. This is surprising to me, but it may be because of the focus of the interviews was on planning aspects of the projects, not on the creation of the product per se. The element of creativity that did appear, however, was the concept of using working software as a means to understand what the real needs are. This is a description of a creative process, in the sense of there being ambiguity and variance that is explicitly used in order to bring forth the final product. This may be quite contrary to the Lean mantra of reducing waste; variance and open-endedness is seen as an asset, not something to eliminate.

The ability – and willingness - to deal with uncertainty, open-endedness and flexibility seem to be a precondition for contractual incompleteness. This is at the

[27] Originally: *"Prioriteten endrer seg hele tiden"* PO, Project A

core of agile practices and comes with the territory of an agile approach. And indeed all the PM and POs mention this as their background or mindset – and the processes used by the four projects are decidedly agile.

5.2.4 Conclusion

What are the circumstances in which contracts can be left deliberately incomplete? There needs to exist trust between the contracting parties. Trust seem has both cultural component and is also strengthened by the reputation the parties have in the market.

In line with theories in contractual incompleteness; the projects are characterized by uncertainty, volatility and to some degree ambiguity and that this is supporting of the low level of up front planning and estimating. Uncertainty and ambiguity combined with the ability to accept change seem to be where contractual incompleteness is most valuable.

5.3 How does contractual incompleteness affect projects?

5.3.1 The absence of discomfort

The effects of estimating or not estimating are maybe the hardest to find in a limited multi-case study, and some effects are more obvious than others and some direct, other indirect. One of the more expected direct findings was discovered in projects B and D. In project B they have recently stopped estimating on one project. Several developers expressed estimating as uncomfortable as mentioned in 4.1.2. So removing estimates also removes something uncomfortable and maybe even threatening from the project.

In project D there was expressed another negative side effect of estimating. As long as the charter and business case is what the project is measured against, they have to report to the steering committee if either cost or benefit changes. At the time of my interviews, the PO and the PM were preparing for a presentation to the steering committee where they had to warn that the estimates were being exceeded. The PO explicitly mentioned this as uncomfortable. The discomfort stems from promising something they have little or nor control over; the volatility and complexity of the domain to a large degree explains this.

I'm not sure whether the absence of discomfort in itself is a goal, but it may have side effects that are to be welcomed. It may represent however an incentive on the part of both team and management to remove estimates from being used for control purposes.

5.3.2 Tracking and change

Another direct effect is that, at least initially, removing estimates means progress is harder to track, especially by the same metrics. Not only is it harder to track, but reducing focus on tracking and documentation is at the very core of agile thinking (Beck et al., 2001). Kanban prescribes cumulative flow diagrams as way of tracking lead-time, throughput and cycle time, but only one of the projects adhere to this (project C).

Project B has removed estimates in one of their projects and the PM find the lack of metrics a challenge. Responding to the challenge, the tasks are made bigger then they would ideally be, as creating several smaller stories would be tracked as an increase in scope. This is an example of "What get's measured, gets done" described in 2.5.5. Still the PM sees the advantages of not estimating as outweighing the problem of tracking.

That said, project D had the numbers they needed to track extensively, but that was still not done largely due to asynchronous organizational processes.

5.3.3 Real-time planning

As discussed in chapter 0, I'm considering the details and timing of specification and their corresponding estimates as the core of a software development contract in our setting. Contractual incompleteness then is about how detailed these specifications are ex ante and how they are (re-) negotiated ex post.

If we look at when planning and estimation is done in these projects, it's done quite close to execution in three of them (A, B and C). The only project that has a rather complete contract with regards to early planning of scope and cost is project D. The biggest positive impact I have observed from less ex ante specification and estimation is the fact that these projects can respond to change, to ideas, to the experience of using the software. This is possible even with estimates, but without the anchoring and confirmation bias an estimate entails, it's certainly easier.

This is particularly evident in project A as it currently uses estimates also for control and scheduling purposes; they try to limit change and volatility.

5.3.4 Seeing is believing

"*When in doubt, divide (the task)!*"[28] PM, Project C

All four projects talk of scoping down as a valuable practice – to reduce scope of a feature to its minimum, to its core. There are several motives for this. One obvious reason is to reduce time to market and start capturing value as early as possible. Even a basic version of a feature or module could give value even if the final, complete version is way off.

Another reason to release a rudimentary product early, is to be able to test a hypothesis or a solution as fast and with as low investment possible. This is the thinking behind the Lean startup "Minimal Viable Product". It isn't necessary to release all the way to the end users in order to achieve this, but you must be able to test in a relevant and realistic way.

As mentioned in 5.2.1, the three least planning projects talk about being able to show the stakeholders and/or the users some working software as a way to build trust:

"They were calmer when we got started. The second or third steering committee meeting maybe, when they started seeing something (aka software)"[29] PO, Project C

---

[28] Originally: *"Hvis du er i tvil, splitt!"* PM, Project C

[29] Originally: *"Roligere når vi først kom i gang med hovedprosjektet. Ved styringsgruppemøte to eller tre, når man begynte å se noe"* PO, Project C

As soon as the customer can actually see, experience and use working software, a lot of uncertainty goes away. Working software becomes a control mechanism in the place of tracking effort hours against estimates. This is what Harris et al. (2009) refer to as Emergent Output Control as presented in 2.4.

Projects need to strike a balance between timing and detail of planning and the need for control (as feedback). One could place the projects of this study and some current practices along these two axes seen in the figure below:

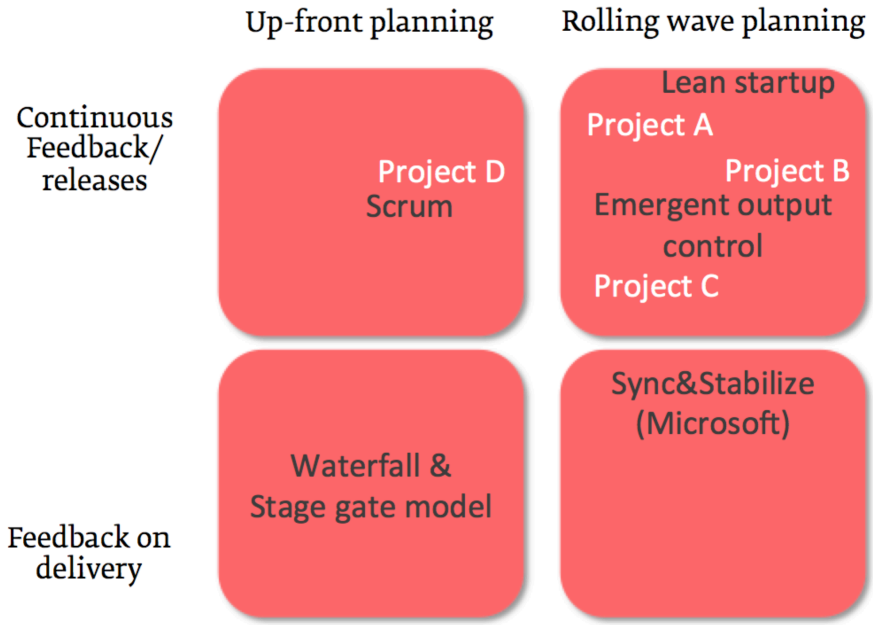

Figure 14 Placement of development methodologies and researched projects by how early they plan and how often they get feedback and/or release running software. T

5.3.5 Process, not construction

In the previous chapter we looked at some advantages of releasing a very minimal version early; capturing value, getting feedback, increasing trust with the stakeholders. But it might be even more novel than that. What if releasing small and early isn't just about driving value creation, but also about understanding the problem itself? Users are able to understand what they need only when they see and can try "something". They produce and release something in order to learn and decide what they need next:

> *One has to kind of find the main thread, end-to-end of what needs to be done, get it into production, then make room for changes as the (real) needs appear. The needs aren't what one thinks, but what appears when you start using it."*[30] PO, Project B

Creating becomes a part of planning – and planning a part of creating. Software development now looks more like an organic, dialectic process rather than incremental construction of a product – the process of understanding is sparked by seeing and using actual running software.

> *"When we deployed to the production environment, it triggered thoughts, and new tasks came up"*[31] PM, Project C

Or to look at it from another angle, there are fewer limitations to this process when you don't have a detailed up-front specification:

"The advantage is that you learn as you go. By not having everything specified in detail up-front, it was easier to focus on what was most important. If you have something there already, it's easier to be colored by it."[32] PO, Project C

This is the effect Brun et al. (Brun & Gjelsvik, 2008) describes as keeping ambiguity to keep options open.

> *"It's when we deliver functionality that the change requests come"* [33] PO, Project D

And in this light, the issue of experience attributes is not a problem anymore. It is a valuable asset – not a bug, but a very valuable feature. Using the experience of a small piece as a way to understand and create the whole is a completely different way of thinking about software development. Software becomes the object of and tool for planning software development (see Figure 15). It utilizes the inherent

---

[30] Originally:*" Må liksom finne hovedtråden, ende til ende-tråden på det som skal gjøres, sette det i produksjon, så ha høyde for å gjøre endringer når behovet viser seg. Behova er ikke det som man tror man har, men det som viser seg når man tar det i bruk"* PO, Project B

[31] Originally: *"Når vi lanserte ut i produksjonsmiljøet, så trigget det tanker, så kom det nye oppgaver"* PM, Project C

[32] Originally: *"fordelene var jo at man lærer underveis. Ved å ikke ha detaljspesifisert ting på forhånd så var det lettere å fokusere på det som var viktigst. Hvis man har noe der fra før av, så er det lett å bli farget av det."* PO, Project C

[33] Originally: *"Det er at når vi leverer funksjonalitet, det er da at endringsønskene kommer."* PO, Project D

ambiguity in the needs of a diverse group of stakeholders and users. Options are kept open and conclusion is deferred in order to learn as much as possible along the way. This is a way of designing and building software that would not be possible in a regime of up-front planning and estimating.

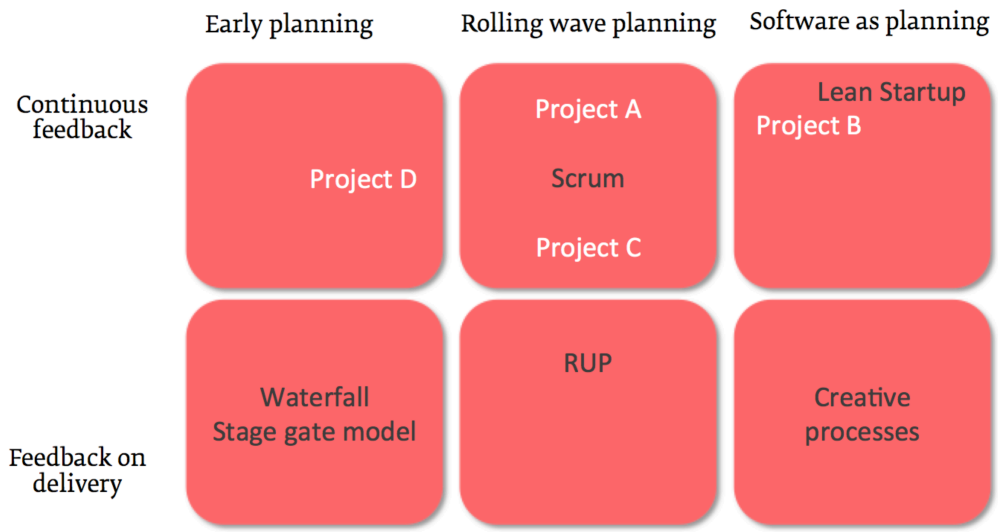

Figure 15 Model to describe temporal aspect of software planning and control practices.

From the projects I have researched it does not seem to be a very deliberate or academic thought process behind this, it seem to be somewhat emergent and based on prior experience of especially the POs. This is one of the organically emergent practices that come into being when you remove the constraints of pre-defined requirements and estimates.

5.3.6 Throwing the baby out with the bath water

The projects all use an approach inspired by Lean manufacturing and Kanban. Although flow, pull based queues and elimination of waste are great benefits, some projects seem to be losing sight of the vision of the project or product they are developing. This to me is one of the most surprising discoveries in the data.

For knowledge workers, having a sense of a common goal and working towards a shared vision is an important motivational factor. It is also considered crucial in order to make the best choices and assumptions in all the little details one encounters in developing even the smallest features every day.

Stray et al (2011) finds the same in studying a Kanban-style software development project:

> *"However, most of the team members in the South project did not know the project plans, nor did they have a clear understanding of the project long-term goals and vision. One developer said: "I don't know how the release goals are defined.""* (Stray et al., 2011)

Another experimental study comparing Kanban to Waterfall and Scrum along nine aspects of project work also finds no evidence of Kanban aiding in "Understanding the whole" (Ikonen et al., 2011). Although Lean and Kanban encourage "seeing the whole", there is no prescribed process or practice for this and it may thus be ignored or skipped.

Abolishing all discussion or presence of high-level vision in order to maximize flow and minimize waste, may be throwing out the baby with the bath water. Projects seem to need to find a way to balance both.

5.3.7 Conclusion

To conclude; what are the effects of removing up-front specifications and estimates from these projects? The two least surprising discoveries are that removing estimates removed discomfort on both the developer and the management side of the project and it reduces the possibility of tracking performance – at least in the traditional way.

A negative effect however seem to be that removing practices for planning and estimating inadvertently has taken away the practices for taking a step back and looking at the big picture.

The most prominent positive effect is the project's ability to learn as they go, and respond to change both in their own knowledge and from external input. Planning in real time allows for all the knowledge and information present to be taken into account and give value.

Frequent releases gives the element of control that the buyer needs in order to ascertain quality and build trust towards the supplier. This again leads to a more novel process; software itself becoming the object of planning, creating and understanding. By releasing minimal increments, ambiguity is left open until users and stakeholders have a better foundation for decision; actual running software. The experience attributes that were considered to be problems are now a project's biggest assets.

5.4 What new practices emerge? And what are needed?

In the absence of traditional planning and design phases in the projects with the least complete contracts (projects A, B and C) there are some practices that emerge and some that are missing and needs to be put in place. Why is this so? Well, in traditional, contractually complete projects, planning processes and stage gates facilitated the need for understanding both what to make and why to make just that. And not only are the traditional practices removed, there is also a different kind of planning and design emerging under these incomplete contracts; a more organic elaboration of software.

To clarify the practices and processes, it's instructive to think of them as three different levels of practice. There needs to be different metaphors or modes to use these different practices. I propose to define it as three levels; an operational, a tactical and a strategic level. As a framework surrounding and enabling these levels are the contracts and the contracting; how do we improve contracting processes and contracts to support these emergent practices?

5.4.1 Operational level: structure and frequent output

On the level of day-to-day operations and getting developments tasks through its workflow, the Lean inspired processes seem to work very well. Teams and management both seem content with the ease of use and low level of formality in these processes. They give the necessary structure – but not more. The Kanban boards offer an accessible visualization of status and progress – and possible clogs and impediments. This is a structured and rational process that is valuable to keep track of operations. It does not, however, support tactical and strategic planning or thinking very well.

5.4.2 Tactical level: organic and emergent planning

At the level of planning, of understanding how the software is to grow, there needs to be support for an emergent process as described in previous chapters. This is not a structured practice, but more of a creative and organic practice where divergence and ambiguity should be allowed to be at play. The early, minimal increments of software are used to understand what their needs *really* are. In such an incremental and organic co-creation, the experience attributes of software is no longer a problem, it becomes an asset as a form of ambiguity that the customer can leverage.

I don't suggest that there should be a separate group of people or a administrative layer, it's more a frame of mind and facilitation of a different kind of collaboration when it comes to thinking about *what* to make.

5.4.3 Strategic level: Seeing the bigger picture

At the level of strategic thinking, there needs to be room for taking a step back and looking at the big picture. My research indicates the need for a (re)newed practice to achieve this. Projects must enable and facilitate situations where the overall mission and vision can be conveyed and management can make sure that all are aligned. There must also exist the opportunity to adjust the goals as the organization gathers knowledge as both internal and external factors shift.

5.4.4 Contracting: Structured approaches to trust

The goal of an incomplete contract, or any contract for that matter, is to maximize value and minimize transaction cost and counterproductive ex post behavior. If we agree that trust and reputation is a prerequisite for having incomplete contracts in software development projects, then that poses a barrier for new partnerships and for new entrants into the market. This is obviously an advantage for the incumbents, but for the overall industry this barrier poses a problem. Could there be practices where these barriers are reduced?

To start with new partnerships, buyers and suppliers that have no previous history of doing business, how could they get to a level of contractual completeness that is adapt to the level of uncertainty, volatility and ambiguity? In our data we have one instance of this. The customer in Project B started with an incentivized fixed price contract to ensure quality and build trust. But is there a way of getting to more lightweight contracts right away? There are some suggested practices in the market; drip funding, result based and Proof of Concept procurement processes.

Drip funding would entail that the supplier only gets paid for small increments, typically the Minimal Viable Products discussed earlier, and that the customer at frequent intervals makes a decision to continue or not. This would shift the risk, at least in part, over to the supplier. I'm not sure whether this model would give the predictability and long-term orientation larger organizations would require, especially on the customer side.

Result based contracts are, to the extent of my knowledge, still to be tested in the Norwegian software development market. With complex domains and deliverables, it could be challenging to determine what parameters to measure that the projects could actually be said to have any control over. This may also end up being extremely complex and "complete" contract, though not on the requirements side, but on the outcome side. You would simply exchange one form of contractual completeness for another.

There are some large government departments who are currently testing Proof of Concept (PoC) procurement processes. The buyer gets several vendors to build a small piece of the intended software in a short time, again a Minimal Viable

Product approach. This serves several purposes. They get to know the supplier resources' actual competency and find out if they can communicate well with them. It also gives input to the choice of technological solution, again software as learning. With a PoC approach, the buyer is able to procure on the basis of what they experience; it gives a better and more realistic picture of the vendors. The buyers still don't get to explore as much details or uncertainty as they might wish, as a short PoC will have to make assumptions and short cuts. It can protect neither the vendor – nor the supplier – against meeting the devil in the details further down the road. But it still gives a lot more value and knowledge than the traditional documentations based procurement process. And it may be forceful enough to build the necessary level of trust prior to contracting and thus to allow for incomplete contracts.

All these three approaches; drip funding, result based contracts and PoC vendor selection, could reduce the barriers to entry, but the barrier would still be a lot higher without prior reputation in the market.

5.4.5 Conclusion

Figure 16 Operational, tactical and strategic levels need to coexist in software development processes

What practices do emerge – and what are needed – in contexts with deliberate contractual incompleteness? There are three levels to address here, as illustrated in Figure 16. On the operational level, there is a need to maintain efficient work processes and enable frequent releases of software.

On a tactical level, there needs to be room for more organic and creative processes for continuous exploration of the organization's and user's needs as software increments keep coming.

And at the strategic level, all involved parties need to have a picture of the whole, the vision and mission of the project at hand.

To enable incomplete contracts in the first place, there is considerable work to be done in speeding up the trust-building processes by different forms of contracting. Since trust is a "time compressing" asset, it represents an advantage for existing players in the market and a barrier to entry to new players.

5.5 Implications for theory

Research on software development tends to be based either on contracting projects (the "buy" option") or complete vertical integration (the "make" option). My research shows examples of a middle ground; there are elements of both integrated resources and resources hired from external consultancies. These hybrid forms of contracting are interesting specimens both for TCE research at large and for the study of software development practices in particular.

The existence of hybrid contracting forms also poses the question of information transference, information stickiness and organizational learning. Contractual incompleteness, dynamic capabilities and organizational learning seem to me to be three theoretical lenses that could shed light on software development research in an interesting way. The design of my study gave too little time and focus to the buying organizations to be able to explore phenomena such as absorptive capacity and organizational learning.

Theory on software engineering, and especially on software estimation, tend to look at software development as a form of construction. What I find in the least planning projects, seem instead to point to software development being far more of an organic, dialectic and even creative process. It would be interesting to see studies that treat software development not as a deviant, immature and non-conforming type of construction, but as a organic process more along the lines of creative processes, product development and innovation.

5.6 Implication for management

5.6.1 For the customer side

Several of the projects express missing the Big Picture. As discussed in chapter 5.4.3, businesses should facilitate communication and internalization of the overall vision of the project at hand.

Customer co-creation and co-location is expressed as a success factor in all the projects. Buyers of software development services must enable and facilitate this in order to create value in the projects but also to retain value and knowledge in the organization in the long run. It is always a challenge to remove skilled and experiences personnel from their normal operations and into projects, but it nevertheless seems a precondition for change taking place during the project life cycle.

Organizations should also consider what sourcing strategies best fit their needs, dynamic capabilities and absorptive capacities. These are classical make-or-buy considerations, but in addition they should take into account the level of up-front specification required. As we have seen through both the literature and the empirical results; a high degree of uncertainty, volatility and complexity calls for a lower degree of contractual completeness – in our terms a less detailed up-front specification and contract. And when more flexibility is left for the ex post renegotiation, customer participation and less formal processes are not only an advantage, but also a necessary capability. It may even be an asset that the buyer can create value from.

There is certainly an element of risk being transferred to the customer in a less complete and rigid contract. This thesis has pointed to several ways of insuring against some of this risk. The risk of opportunism could be counteracted with reputation, closeness and also more incentivized models. The risk inherent in services with experience attributes, as suggested in Harris et al. (2009) and shown in the empirical results, is mitigated by having frequent and early releases to ensure feedback.

5.6.2 For the supplier side

The warning of not loosing sight of the Big Picture also goes for the supplier side. To the extent that the supplier is shaping the projects methodology (as is the case in half of the projects in my sample), the responsibility – and possibility to change – the way this is done rests on the supplier. The challenge is to be able to balance practices for looking at the big picture while at the same time keeping processes that are optimal for continuous and flowing software development work without too much administrative overhead and context switching. One needs to be able to

move between the operational and the strategic level according to the needs of the project and context at hand.

As trust and reputation are shown to be important preconditions, or even enablers, of incomplete contracts, work towards building and signaling reputation is indeed important. The same goes for strengthening existing relationships and focusing on maintaining trust and collaboration. Contracts that are less complete and more built on trust and co-creative processes require discipline and professionalism from the supplier. With a more complete contract, there are explicit and clear rules to govern the contracted work. On the other hand, shading and counterproductive behavior is mentioned also as a risk in complete contracts; the supplier and customer may start abiding by the letters of the contracts instead of by its intention. A more incomplete contract requires that the supplier live up to that trust every day. Trust takes time to build, but can be torn down fast.

The supplier side should also experiment and welcome new models for contracting; drip funding, result based and PoC-driven contracting.

As we have seen in two of the projects, there is a hybrid of multi-sourcing and in-sourcing, a wedge between "make" and "buy". This might be a challenge to software development consultancies that are based on selling entire project deliverables and project teams, but more adapt to companies who rest more on selling single resources. Whether this is then a threat or an opportunity depends on the supplier's business model.

5.7 Implication for policy

Public procurement processes have rules to ensure impartiality and anti-corruption that can hinder customer co-creation in the early phases of a project. Public procurement also has a tendency to rely on rigid and vast requirements specifications for choice of vendor and contracting. This tends to mean highly complete contracts and a costly and extensive administrative layer. This, as we have seen, is not a god fit in complex, volatile and uncertain circumstances. This realization needs to be reflected in the public procurement regulations.

There also needs to be developed better contract standards. At the core of most software standard contracts are the idea of detailed specifications and the supplier is asked to give a suggested solution and give an estimate. There are some changes to this in the making; there are two "agile" software development contracts in use in Norway today; PS2000 Smidig and PS 2000 SOL.

5.8 Limitations and further research

The sample in this study is from one consultancy only; they might rest on the same frame of reference and the same view on methodology and contracting. For the sake of this study, the supplier proved a good place to find sample since this topic is little researched and the practice of not estimating rather new. It would be very interesting to conduct the same type of research with a wider set of supplier firms to see what variables that introduces.

There are no fixed price projects in my sample, would they look very different? I'm guessing they would. Project B was initially a hybrid/incentivized fixed price contract, but had much the same make-up as it's current incarnation. On the topic of contracting and contracts it would be interesting also to study emergent contracting models as mentioned in chapter 5.4.4.

There is no cultural diversity in my sample, and the Norwegian culture is considered to be high on societal trust. It would be interesting to do more research on cross-cultural projects and projects entirely based in other countries with different cultural characteristics.

To get a better answer to what contractual incompleteness does to projects, it would be valuable to follow two comparable projects over a longer period of time and in more depth. Especially a semantic analyses would be interesting to shed light on how customers, supplier, team members and stakeholders talk about and relate to planning, scope and control over some time. A longitudinal study could also explore the aspects of dynamic capabilities and organizational learning in the buying organization.

There is also need to do more quantitative research. Based on my findings, it would be interesting to look for correlations between the different aspects discovered; contracts, uncertainty, ambiguity, trust, reputation, information stickiness and organizational learning. Another interesting quantitative study would be to compare up-front requirements with what features were actually developed and what features in the system that are actually used when it's in production. This could give a picture of how much and what kind of change actually takes place - from an initial idea, through development and into actual use.

6 Conclusion

The managerial dilemma of entering into a contract without being able to ascertain the costs it will incur is at the heart of this thesis. We have established some mechanisms that seem to counteract this dilemma; trust, reputation, cooperation and frequent feedback.

So when is it worthwhile entering into this dilemma, when is it valuable and possible to leave contracts deliberately incomplete? Both the data and the literature show us that in contexts with uncertainty, complexity and ambiguity this seems to be a rational and valuable choice due to the need for ex post changes. Delivering products and services with experience attributes further enhances this value; it is difficult to describe and ascertain quality up front. Both the literature and my data point to that trust is also an important precondition for incomplete contract to be possible.

So, how does contractual incompleteness influence projects? Frequent releases and short feedback loops seem to work at control mechanisms. Frequent releases combined with planning and elaboration of software specification on the basis of these releases seems to emerge as an organic and creative process where software itself becomes the object of planning. This opens up for a different way to look at software; not as an immature form of contracting, but a creative and organic endeavor more in line with explorative product development and creative processes.

A negative side effect, however, seems to be that the projects with low contractual completeness need to be vary of not losing touch with the big picture. Looking up at the mission and vision at a strategic level is important both for the sake of team motivation and their ability to make the right decisions and assumptions in all their daily work.

These findings point to three distinctly different modes of working; a structured operational mode, an organic and creative tactical mode and a more visionary strategic mode.

How do these practices influence governance and contracting in both buyer and supplier organizations? There seem to be some interesting hybrid forms of contracting that have elements of both buying and vertical integration. How the organizations choose to procure seem to be dependent on how they build their own specific resources and capabilities, how they think about information stickiness. The simple dichotomy of make-or-buy might be too simplistic. Enabling co-creation and emergent exploration may be a more central question than the organizational structure of procurement.

7 Bibliography

Abdel-Hamid, T., & Madnick, S. (1986). Impact of schedule estimation on software project behavior. *IEEE Software*, *3*(July).

Anderson, D. J. (2013). *Kanban*. Blue Hole Press Inc.

Anderson, E., & Gatignon, H. (1986). Modes of foreign entry: A transaction cost analysis and propositions. *Journal of international business studies*.

Bahli, B., & Rivard, S. (2003). The information technology outsourcing risk: a transaction cost and agency theory-based perspective. *Journal of Information Technology*, *18*(3), 211–221. doi:10.1080/0268396032000130214

Bajari, P., & Tadelis, S. (2001). Incentives versus transaction costs : a theory of procurement contracts. *RAND Journal of Economics*, *32*(3), 387–407.

Beck, K., Cockburn, A., Martin, R. C., & Schwaber, K. (2001). The Agile Manifesto. *agilemanifesto.org*.

Bernheim, B., & Whinston, M. (1998). Incomplete Contracts and Strategic Ambiguity. *American Economic Review*, *88*(4), 902–932.

Brun, E., & Gjelsvik, M. (2008). Benefits of Ambiguity In New Product Development. *International Journal of Innovation and Technology Management*, *5*(3), 303–319.

Bryman, A., & Bell, E. (2007). *Business Research Methods*. *Library* (Vol. 2nd, p. 786). doi:10.1.1.131.2694

Caballero, R. J., & Hammour, M. L. (1996). The "Fundamental Tranformation" in macroeconomics.

Cohn, M. (2005). *Agile Estimating and Planning*. Prentice Hall.

Cohn, M. (2012). An Introduction To Scrum. *Mountain Goat Software*. Retrieved December 18, 2013, from http://www.mountaingoatsoftware.com/presentations/an-introduction-to-scrum/

Cusumano, M. (2004). *The Business of Software* (1st ed.). New York City: Free Press.

Eisenhardt, K. (1989). Building Theories from Case study research. *Academy of management review*, *14*(4), 532–550.

Fehr, E., Hart, O., & Zehnder, C. (2011). Do Informal Agreements and Renegotiation Shape Contractual Reference Points? *Draft*.

Fink, L., Lichtenstein, Y., & Wyss, S. (2013). Ex post adaptations and hybrid contracts in software development services. *Applied Economics*, *45*(32), 4533–4544. doi:10.1080/00036846.2013.791021

Grimstad, S., & Jørgensen, M. (2007). Inconsistency of expert judgment-based estimates of software development effort. *Journal of Systems and Software*, *80*(11), 1770–1777. doi:10.1016/j.jss.2007.03.001

Halkjelsvik, T., Rognaldsen, M., & Teigen, K. H. (2012). Desire for control and optimistic time predictions. *Scandinavian journal of psychology*, *53*(6), 499–505. doi:10.1111/j.1467-9450.2012.00973.x

Halonen-Akatwijuka, M., & Hart, O. (2013). More is Less: Why Parties May Deliberately Write Incomplete Contracts. *Draft*, (April).

Harris, M. L., Collins, R. W., & Hevner, a. R. (2009). Control of Flexible Software Development Under Uncertainty. *Information Systems Research*, *20*(3), 400–419. doi:10.1287/isre.1090.0240

Hippel, E. Von. (1994). "Sticky Information" and the locus of problem solving: Implications for Innovation. *Management science*, 429–439.

Hofstadter, D. (1979). *Gödel, Escher, Bach: An Eternal Golden Braid* (p. 777). Vintage.

Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P., & Abrahamsson, P. (2011). On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation. *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*, 305–314. doi:10.1109/ICECCS.2011.37

Jeffery, D. R., & Lawrence, M. J. (1985). Managing programming productivity. *Journal of Systems and Software*, *5*(1), 49–58. doi:10.1016/0164-1212(85)90006-8

Johnson, J. H., & Yarmouth, W. (2001). CHAOS report 2000. *Standish Group*, 132–135.

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, *70*(1-2), 37–60. doi:10.1016/S0164-1212(02)00156-5

Jørgensen, M. (2013). The influence of selection bias on effort overruns in software development projects. *Information and Software Technology*, *55*(9), 1640–1650. doi:10.1016/j.infsof.2013.03.001

Jørgensen, M., & Moløkken-Østvold, K. (2004). Reasons for software effort estimation error: impact of respondent role, information collection approach, and data analysis method. *Software Engineering, …*, *30*(12), 993–1008.

Jørgensen, M., & Moløkken-Østvold, K. (2006). How Large Are Software Cost Overruns? A Review of the 1994 CHAOS Report. *Information and Software Technology*, (1325).

Jørgensen, M., & Sjøberg, D. (2001). Impact of effort estimates on software project work. *Information and Software Technology*, *43*(15), 939–948.

Kahneman, D. (2011). *Thinking Fast and Slow*. Penguin.

Killick, N. (2013). http://neilkillick.com/. *Blog*.

Koh, C., Ang, S., & Straub, D. W. (2004). IT Outsourcing Success: A Psychological Contract Perspective. *Information Systems Research*, *15*(4), 356–373. doi:10.1287/isre.1040.0035

Lacity, M. C., Willcocks, L. P., & Khan, S. (2011). Beyond Transaction Cost Economics: Towards an endogenous theory of Information Technology Outsourcing. *The Journal of Strategic Information Systems*, *20*(2), 139–157. doi:10.1016/j.jsis.2011.04.002

MacCormack, A., & Verganti, R. (2003). Managing the sources of uncertainty: Matching process and context in software development. *Journal of Product Innovation …*, (617), 217–232.

Mishra, D., Heide, J., & Cort, S. (1998). Information Asymmetry and Levels of Agency Relationships. *Journal of marketing Research*, *35*(3), 277–295.

PMI. (2013). What is Project Management? *pmi.org*.

Poppendieck, T., & Poppendieck, M. (2013). The Lean Mindset. *Poppendieck.com*. Retrieved December 27, 2013, from http://www.poppendieck.com/

Ries, E. (2011). *The Lean Startup*. Random House.

Rousseau, D. M. (1989). Psychological and implied contracts in organizations. *Employee Responsibilities and Rights Journal*, *2*(2), 121–139. doi:10.1007/BF01384942

Senapathi, M., & Srinivasan, A. (2011). Understanding Post-Adoptive Agile Usage -- An Exploratory Cross-Case Analysis. *2011 AGILE Conference*, 117–126. doi:10.1109/AGILE.2011.19

Stray, V., Moe, N., & Dingsøyr, T. (2011). Challenges to Teamwork: A Multiple Case Study of Two Agile Teams. *Agile Processes in Software Engineering …*, 146–161.

Susarla, A., Barua, A., & Whinston, A. B. (2009). A Transaction Cost Perspective of the "Software as a Service" Business Model. *Journal of Management Information Systems*, *26*(2), 205–240. doi:10.2753/MIS0742-1222260209

Tabrizi, B. N., & Eisenhardt, K. (1995). Accelerating Adaptive Processes : Product Innovation in the Global Computer Industry. *Administrative Science Quarterly*, *40*(1), 84–110.

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, *185*(4157), 1124–1131.

Williamson, O. E. (2005). The Economics of Governance. *American Economic Review*, *95*(2), 1–18. doi:10.1257/000282805774669880

Xue, M., & Field, J. M. (2008). Service Coproduction with Information Stickiness and Incomplete Contracts: Implications for Consulting Services Design. *Production and Operations Management*, *17*(3), 357–372. doi:10.3401/poms.1080.0024

Yin, R. (1984). *Case Study Research*. (Sage, Ed.). Beverly Hills.

Zuill, W. (2013). Life, Liberty, and the Pursuit of Agility. Retrieved December 27, 2013, from http://zuill.us/WoodyZuill/

8 Appendix

8.1 Appendix A: Interview guide

Intro

Det jeg ønsker å snakke med deg om i dag, er planlegging og styring i ditt prosjekt <…>.

Med planlegging mener jeg hvordan dere jobber med omfang, kost, risiko, kvalitet og tid framover.

Med styring er jeg mest opptatt av hvordan dere måler fremdrift, leveranse og kost i prosjektet. Jeg er interessert i veien frem til den måten dere jobber på i dag og hvordan dagens situasjon oppleves.

For å kunne registrere innspillene jeg får, ønsker jeg å ta opp samtalen. Går det bra for deg? Din anonymitet blir ivaretatt og det vil ikke være mulig å spore svarene tilbake til deg eller til ditt prosjekt.

Praktisk

- Prosjekt
- Rolle
- Relasjon med kunden i antall år
- Demografi (alder, kjønn, senioritet, erfaring i rollen)
- Sted
- Dato

Deskriptiv

Hvordan ble prosessen til?

- Hvilke valg var allerede gitt? Av kontrakt, av sedvane, av lovkrav f.eks.
- Hvilke prosesser kunne dere påvirke?
- Hvem bidro til å påvirke?
- Hvorfor ble prosessen slik den er i dag?

Kan du beskrive for meg hvordan planlegger dere i dag?

- Gjerne konkret og detaljert
    - Situasjoner, settinger, lokasjon, verktøy
    - Detaljering av funksjonalitet
    - Begrep av omfang (scope boundaries)
    - Feedback mechanisms/frequency

- Hvem er involvert
- Konkrete praksiser (planleggingsmøter, stand-up-møter etc)

Hvis estimater:

Hvordan estimerer dere?

- Hvem bidrar?
- Hvordan forholder dere dere til sannsynlighet, usikkerhet, uklarheter?
- Beskriv gjerne prosessen konkret og detaljert for meg (når, hvor, artefakter, verktøy)

Hva brukes estimatene til?

- Hvor registreres estimatene?
- Hvem vet om dem?
- Brukes det til å tracke faktisk fremdrift mot?
- Måleenhet for estimater og faktisk pådrag?

Hvis ikke estimater:

Alternativ praksis, hvordan håndterer dere kost/nytte/måling?

- Hvordan holder dere styr på kost?
- Hvordan holder dere styr på fremdrift?
- Rapportering?
- Hvorfor estimerer dere ikke?
- Er det noe spesielt som gjør det mulig i dette prosjektet?
- Hvordan påvirker det prosjektet?

Opplevelse av prosess

Hvordan fungerer planleggingsprosessene i prosjektet?

- Hvordan liker <andre roller> måten dere jobber på?
- Hva fungerer godt?
- Hva er mest utfordrende?
- Hva fungerer ikke?
- Er det planlagt større endringer i måten dere planlegger og styrer på?

Hva slags kontrakt opererer prosjektet under?

- I hvor stor grad dikterer kontrakten måten dere jobber på fra dag til dag /med planlegging
- Var du meg på å utarbeide kontrakten, velge kontraktsform?
- Noen tanker om kontraktsformen dere har? Hindrer den dere på noen måte? Er den hjelpsom på noen måte?

Hva tror du din motpart forventer av deg (særlig mtp planlegging og styring)?

Avslutning, se fremover

- Er det noe rundt estimering, planlegging og styring som du har kommet på i løpet av samtalen?
- Hvordan ønsker du å videreutvikle prosessen i ditt prosjekt videre de nærmeste månedene?
- Hva ville være din ideal-prosess?
- Hvordan tror du prosesser for prosjektstyring av softwareprosjekter er om 10 år?

Postskript

Hvordan opplevdes intervjuet?

- Setting
- Stemning
- Engasjert intervjuobjekt?
- Spesielle hendelser underveis
- Ble det tegna, doodla eller brukt andre måter å formidle på under intervjuet (dokumentér!)