Information and Software Technology 55 (2013) 2154–2171

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

When agile meets the enterprise

Guus van Waardenburg [b], Hans van Vliet [a,*]

[a] VU University Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
[b] Avanade Netherlands, Versterkerstraat 6, 1322 AP Almere, The Netherlands

ARTICLE INFO

Article history:
Received 20 August 2012
Received in revised form 24 July 2013
Accepted 25 July 2013
Available online 17 August 2013

Keywords:
Agile development
Enterprise environment
Grounded theory

ABSTRACT

Context: While renowned agile methods like XP and Scrum were initially intended for projects with small teams, traditional enterprise environments, i.e. environments where plan-driven development is prevalent, have also become attracted by the promises of a faster time to market through agility. Agile software development methods emphasize lightweight software development. Projects within enterprise environments, however, are typically confronted with a large and complex IT landscape, where mission-critical information is at play whose criticality requires prudence regarding design and development. In many an organization, both approaches are used simultaneously.
Objective: Find out which challenges the co-existence of agile methods and plan-driven development brings, and how organizations deal with those challenges.
Method: We present a grounded theory of the challenges of using agile methods in traditional enterprise environments, based on a Grounded Theory research involving 21 agile practitioners from two large enterprise organizations in the Netherlands.
Results: We organized the challenges under two factors: *Increased landscape complexity* and *Lack of business involvement*. For both factors, we identify successful mitigation strategies. These mitigation strategies concern the communication between the agile and traditional part of the organization, and the timing of that communication.
Conclusion: Agile practices can coexist with plan-driven development. One should, however, keep in mind the context and take actions to mitigate the challenges incurred.

1. Introduction

In the last ten years the use of agile methods has become widespread [26]. While renowned methods like XP and Scrum were initially intended for projects with small teams, traditional enterprises have also become attracted by the promise of a faster time to market for their projects [9,48,16]. Agile software development methods, particularly Scrum, emphasize lightweight software development: the art of maximizing the amount of work not done. However, projects within enterprise environments are typically confronted with large and complex IT landscapes, where mission-critical information is at play whose criticality demands utter prudence regarding design and development.

Plan-driven methods emphasize "a rationalized, engineering-based approach" [79] in which it is claimed that problems are fully specifiable and that optimal and predictable solutions exist for every problem. Advocates of plan-driven methods recommend rigorous planning, codified processes, and extensive reuse to make software development an efficient and predictable activity. *Plan-driven Software Development*, when referred to in this paper, encompasses all software development methodologies characterized by a structured software life-cycle consisting of predefined phases, extensive design and requirements documentation, and hierarchical organization structures. Plan-driven development comes with a low release frequency, typically one release per six or more months. Plan-driven development also is contract-driven: the requirements serve as a contract. Changes in requirements often are not dealt with until a next release. Whenever the adjective "traditional" is used in the remainder of this article, as in "traditional team" or "traditional requirements elicitation", it refers to an organization or practice typically associated with a plan-driven approach to software development. Organizations that employ plan-driven development often have a centralized IT department that provides all IT services [4]. By implication, budget responsibility for IT resides in this IT department. This IT department generally uses a traditional project structure, as defined in a project management method such as PRINCE2 [59]. The non-IT part of the organization is referred to as "business" (as in "business representatives", for example). In this paper, we use the term *Traditional Enterprise* to refer to an organization that has the above characteristics.

* Corresponding author. Tel.: +31 20 598 7768; fax: +31 20 598 7728.
E-mail addresses: guusvanwaardenburg@gmail.com (G. van Waardenburg), hans@cs.vu.nl (H. van Vliet).

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.07.012

In contrast to plan-driven software development, agile software development methods address uncertainty:

> "*the continual readiness of an ISD [Information Systems Development] method to rapidly or inherently create change, pro-actively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment.*" —Conboy, 2009 [23].

Agile Software Development, when referred to in this paper, encompasses software development methodologies characterized by a continuous readiness to rapidly realize change, pro-actively or reactively embrace change, and learn from change while contributing to customer value. In particular, Scrum, the development methodology used in the environments studied (see also Section 2.3), fits this definition. A focus on working code right from the beginning, delivery cycles of 2–4 weeks (so-called Sprints), and having business representatives on the team are but three practices to achieve the above benefits [69].

There are many software development methodologies that fall in between plan-driven development and agile development, and exhibit several of the characteristics of agile development. Examples include incremental development, prototyping, and DSDM. This continuum of methodologies is discussed in Abrahamsson et al. [1], Boehm [12], and textbooks like van Vliet [79].

Increasingly, agile practices are being applied in traditional enterprises as well. This may take many forms. Balance studies like [13] focus on choosing the "right" point on the plan-driven – agile continuum for a project. For example, one might decide to have an upfront architectural phase to decide on the overall structure of a system, followed by a series of Scrum sprints. The result is a project which exhibits a mixture of plan-driven and agile aspects. There are many such intermediate forms feasible [13]. Our research took place in organizations where there was a relatively strict separation between the traditional IT department and the agile projects. In these organizations, most projects follow a plan-driven approach, while some follow an agile approach. Note we do not want to suggest that agile is good and plan-driven methods are bad; such is a false dichotomy [49].

In this paper, we present a grounded theory of agile practices in traditional enterprises, based on a Grounded Theory research involving 21 agile practitioners from two large enterprise organizations in the Netherlands. Our focus is the co-existence of both agile and plan-driven development practices within the same enterprise, not the migration from a plan-driven organization to an agile one, as for instance discussed in Laanti et al. [48], Nerur et al. [58], Petersen and Wohlin [63], or the balance between plan-driven and agile development, as discussed in for instance Boehm and Turner [13], Magdaleno et al. [52]. The theory developed consists of two factors, *Increased IT landscape complexity* and *Lack of business involvement*, that are a result of the introduction of agile practices in an enterprise context. For both factors, the theory identifies the underlying causes as well as its consequences, and successful ways to cope with the corresponding challenges.

A large and complex IT landscape (i.e. the way IT is organized, in combination with the portfolio of systems and their interdependencies – see also Section 2.3) is inherently connected to enterprise software development. Years of evolution of this landscape to support the business results in a collection of highly interconnected information systems. This landscape complexity is increased when, in addition to plan-driven development projects, agile projects are being launched. In our study, coping with the complexity that results from having both plan-driven and agile development emerged as one of the most daunting tasks for agile practitioners. A detailed description of *Increased IT Landscape Complexity* and its characteristics (the first part of our theory) can be found in Section 3.

During our research lack of business involvement was indicated as *"one of the hardest things to cope with when practising agile"*. Many aspects of business involvement are similar to findings about customer involvement or customer collaboration as observed in previous studies. A detailed description of *Lack of Business Involvement* and its characteristics (the second part of our theory) can be found in Section 4.

Our results on the major challenges and its causes and consequences are in line with, and corroborate, findings of earlier studies, most of which concern the introduction of agile methods, or the transition from plan-based to agile methods, rather than the simultaneous use of both. This need not come as a surprise since, during a transition from plan-based development to agile methods, both will temporarily co-exist, and likewise will the challenges this brings. However, the co-occurrence of agile practices and plan-driven development requires one has to find ways to cope with these challenges. The contribution of our research is in identifying successful mitigating actions for doing so. These mitigating actions serve to support the communication between the agile and the traditional part of the organization, and the timing thereof. They serve as boundary spanning activities [50] between the two communities.

The discussion of our main findings in Sections 3 and 4 is preceded by a description of the research design in Section 2, and followed by a discussion of our findings in the context of related work in Section 5. We conclude the paper with a discussion of threats to validity in Section 6, and conclusions in Section 7.

2. Research design

As research method, we opted for Grounded Theory. The Grounded Theory Method (GTM) aims to develop a theory from data rather than gather data in order to test a theory or hypothesis.

The Grounded Theory Method was selected as method for this research for the following reasons: first and foremost, agile development focuses on individuals and interactions [8], and GTM allows the study of social interaction and behavior [33,60]. Secondly, GTM enables theory generation instead of extending or verifying existing theories [33]. Thirdly, GTM is particularly useful when researching relatively new areas or to approach well-known areas from a new point of view; and there has been little research on agile development within traditional enterprises. Finally, GTM has been used successfully and its popularity as a research method within the fields of Software Engineering and Information Systems is ever-growing [22,2,3,10,40]. A selection of GT practices was applied during this research; Table 1 shows a brief description of each technique used. Our approach can be classified as "evolved GTM" or "Straussian", rather than "classic GTM" or "Glaserian", following the typology of Matavire and Brown [53].

2.1. Start-up and selecting research area

In order to effectively study and uncover the main problems of the participants, classic GTM recommends refraining from formulating a research problem or a question up front [32]. The rationale behind this recommendation is that (a) GTM is meant to generate new theory, and having a preconceived research problem can cause the researchers to be limited in their explorations; and (b) the research problem should be the problem of the participants under study and should not be preconceived or forced, rather it should be allowed to emerge [32]. In evolved GTM, it is more common to do an a priori investigation and literature review to provide context.

Table 1
Applied grounded theory techniques.

Term	Description
Minor Literature Review	A minor literature review is conducted – just enough to hold a substantive conversation with the participants. In this research, the initial conversations with experts aided in the selection of initial literature
Open Coding	The first step of the analysis process. The researcher extracts key-points from raw data and subsequently assigns a code (i.e. a phrase that summarizes the key-point in two or three words [5])
Constant Comparison Method	The process of constant comparing codes emerging from the data against codes from the same interview, and those from other interviews, resulting in a higher level of data abstraction [33]
Memoing	The process of writing out theoretical notes. Memo's are written throughout the GT process in a ongoing fashion. The researcher captures the conceptual links between categories by writing down his reflections on different categories
Core Category	During the GT process, several categories emerge from the data analysis. The category that accounts for most variations in the data and meaningfully relates with other categories is called the core category [33]
Theoretical Saturation	When data collection on a certain category reaches a point where further analysis does not result in new insights, the category has reached *theoretical saturation* [33]. At this point, data collection and coding can be halted for this particular category
Major Literature Review	At the point of an emerging theory, an elaborate literature review is conducted. The purpose of this review is to identify how existing literature relates to the emerging theory
Write-up	Finally, the resulting theory is written up, following the theoretical outline resulting from the research

Our study was initiated by a consultancy firm. This consultancy firm was interested in increasing its knowledge about IT challenges within the companies they work for.

We started with an exploratory phase. Open interviews with experts from the consultancy firm were held and a shallow literature study was conducted to surface potential interesting topics. Themes like *Enterprise Environments*, *Agile Software Development*, *Enterprise Architecture* and *Lean Architecture* were discussed and a selection of relevant literature was studied.

The subject *Enterprise Agile* was selected to represent the overall interest within the consultancy firm as well as the companies they work for. As one expert said: *"Our clients - the big enterprises that is – have heard of agile and its promises to drastically shorten the time to market. They are very interested in the concept, but remain hesitant to implement agility in their organization. Their main concern is the loss of predictability, they need certainty as there is a load of mission-critical information at stake in their projects. How to go about architecture? What about documentation? Who can we blame when business value isn't met, when we don't have agreements clearly written down? These are difficult questions we need to answer."*

The researcher may be advised to not formulate a research question up front, he is however required to choose a general area of interest. We started by exploring *Enterprise Agile* as an area of research, mainly due to the aforementioned value expressed. There were many misconceptions about the adoption of agile methods within enterprises. Along the way the social and human aspects of agile adoption within plan-driven environments began to draw our interest.

2.2. Selecting contexts and research proposal

'Early adopters' do exist, having implemented agile processes within their organization. Several categories can be distinguished, from the enterprises that implemented agility throughout their whole organization to enterprises that implemented agile processes alongside their traditional project organization.

Our research focusses on the latter category. Many enterprise organizations date from before the advent of agile methods, and have IT departments that originated when only plan-driven development methods were in vogue, and these IT departments tend to still use those methods. Moreover, phenomena related to the adoption of agile processes within enterprise organizations should be more obvious within organizations that are not used to agility, as opposed to organizations that have been agile since their start-up. A selection of companies that satisfied this criterion was approached. As a result two companies showed interest in the research proposal and agreed to participate. The remainder of this section discusses the research context and the application of the GTM in our research.

2.3. Research context

This research is based on interviews with 21 participants from two enterprise organizations. The first author did all interviews. The duration of the interviews was 1–1, 5 h, and each participant was interviewed once, and separately. The study was conducted between September 2011 and March 2012. Company A is an international financial corporation with 26,000 employees, 10,000 of which are based in The Netherlands; Company B is a large international manufacturing corporation with over 100,000 employees, 14,000 of which are based in The Netherlands. Interviews were first held with participants from Company A. After initial coding and categorization, participants from Company B were interviewed, and their data was used to validate and deepen the codes and categories. The following paragraphs describe some essential characteristics of the IT landscapes under scrutiny.

2.3.1. Centralized IT department

Both organizations have a centralized IT department, meaning that IT services are provided by one specialized department. In both organizations too, budget-responsibility resides with the IT department.

2.3.2. Traditional project organization

The IT organizations of both companies studied are hierarchical in nature. They both employ a traditional project structure in accordance to the PRINCE2 methodology.

2.3.3. Scrum practices

All of the following Scrum practices listed below (see Schwaber and Sutherland [69]) were implemented at the two organizations:

- *Scrum roles*: The Product Owner, the Scrum Master and the Development Team.
- *Scrum events*: The Sprint, Sprint Planning Meeting, Daily Scrum, Sprint Review, Sprint Retrospective.
- *Scrum artefacts*: Product Backlog, Sprint Backlog and Increment.
- The definition of Done is prepared.

2.3.4. Front-end development

Most participants are part of an agile team developing front-end software, meaning the development of UI and closely related software, mainly web-based. Multiple languages were mentioned during the research, among others: HTML, CSS, and JavaScript.

2.3.5. Back-end development

As opposed to front-end development, back-end development includes databases, services, and maintenance/evolution of legacy software. Back-end integration is an important aspect for all participants. All information resides in back-end systems, forcing the

front-end teams to interface with systems of different compositions and types. Languages and tools that were mentioned for the development of backend systems are, among others: COBOL, Java, and Oracle.

2.3.6. Projects involved

At company A, participants from four projects were involved in our investigations. These projects were all part of one bigger project. All teams consisted of approximately 10 persons. Three of these teams worked on front-end development, one team worked on the back end and was responsible for realizing integration with legacy systems. The average duration of these projects was one year for the first phase, and one year for the second phase. Subsequent phases had not been planned yet. At company B, three projects were involved. These were part of a bigger project too, and each of those teams consisted of 8–10 people. The duration of these projects was 0,5–1 year. So the projects were comparable: they were roughly of the same size, had a similar duration, all used Scrum, and all were concerned with developing front-end functionality on behalf of large legacy back end software. At Company A, the goal was to achieve a seamless integration with their financial services. At Company B, the goal was to let business units inform clients about the company's products. In both companies, the systems were thus intended for use by outside clients. At Company A, the backend system was outsourced. This was partly so at Company B (when interfaces between backend and frontend had to be developed, developers from the backenmd system were added to the frontend team).

2.3.7. Participants

Most participants are members of agile teams developing front-end software. We interviewed participants from each of the seven agile teams mentioned above (this is indicated in Table 2). Other participants represent back-end parties or the programme as a whole. At both companies, the participants had experience with software development for three or more years. The experience with agile was one year or more. Table 2 gives an overview of all practitioners involved in this research.

2.3.8. Collocation

At company A, the different teams had their office in different buildings, though each team was collocated. At company B, most agile teams shared a single large office space.

2.3.9. History of agile

In both companies studied, the agile initiative started bottom-up, i.e. the development teams are the first in the organization to implement agile practices. Company A had been using agile methods since one year, Company B since three years. In both organizations, agile methods were by and large only known to the IT department, and the business had little to no knowledge thereof. Fragmentation of decision authority as well as changes in management hampered the process of company-wide adoption of agile methods.

2.4. Data collection

Data collection in GTM is guided by Theoretical Sampling, which is *"the process of data collection for generating theory whereby the analyst jointly collects, codes, and analyses data and decides what data to collect next and where to find them, in order to develop his theory as it emerges."* [30]. Due to time and access constraints theoretical sampling could not be fully implemented in the process (see also Section 6).

All interviews were conducted in a semi-structured fashion, allowing new questions to be brought up during the interview as a result of what the interviewee says. After evaluating and coding the interviews, some questions and their answers were discarded to focus on the main categories that emerged from the analysis process.

2.5. Data analysis

Data analysis in a Grounded Theory (GT) study is called coding. We adapted our data analysis process from Hoda et al. [39]. The theory is generated through several levels of abstraction.

The first step in data analysis is called open coding [33]. Key points are derived from interview quotations. These are then assigned one or more codes – a phrase that summarizes the key point in two or three words. For example:

> Interview quotation: "*Business needs to take into account the environment. It's not just one website which has to be realized. A typical change has repercussions on many levels. Within our process, but outside it as well. We can't 'hack' in every change. We need to take into regard the stakes of all parties involved. We have a 2 weekly scheduled meeting where we analyse impact of changes organization wide. Can we tackle this change directly? If not, we put it on the product backlog for further analysis.*" —P1, Scrum Master

Table 2
Participants.

P#	Position	Method	Company/team	Team size	Iter. (weeks)
P1	Scrum Master	Scrum	Company A – Team 1	12	3
P2	Product Owner	Scrum	Company A – Team 1	12	3
P3	Project Manager	Scrum	Company A – Team 2	12	3
P4	Developer	Scrum	Company A – Team 1	12	3
P5	Product Owner	Scrum	Company A – Team 2	12	3
P6	Business Analyst	Scrum	Company A – Team 3	10	3
P7	Project Manager	Scrum	Company A – Team 3	10	3
P8	Developer	Scrum	Company A – Team 3	10	3
P9	Scrum Master	Scrum	Company A – Team 3	10	3
P10	Operational Manager	N/A	Company A	N/A	N/A
P11	Product Owner	Scrum	Company A – Team 4	10	3
P12	Release Manager	Waterfall	Company A	N/A	N/A
P13	Scrum Master	Scrum	Company B – Team 1	8	1
P14	Scrum Master	Scrum	Company B – Team 2	10	3
P15	Product Owner	Scrum	Company B – Team 2	10	3
P16	User Experience Expert	Scrum	Company B – Team 1	8	1
P17	Business Stakeholder	Scrum	Company B – Team 3	N/A	N/A
P18	Project Manager	Scrum	Company B – Team 3	10	3
P19	Project Manager	Scrum	Company B – Team 2	10	3
P20	Application Manager	N/A	Company B	N/A	N/A
P21	Product Owner	Scrum	Company B – Team 3	10	3

This quotation next leads to the following key points and codes:

Key Point: "*Changes can influence multiple parties involved, this makes it hard to just hack in any change.*"

Codes: *multiple parties involved; influence by changes* (P1)

By constantly comparing codes from an interview against other codes of that same interview and those from previous interviews, we grouped codes to produce a higher level of abstraction, called concepts in GTM. This is called the *constant comparison method* [33]. For the above example, this would lead to:

Concept: *changes have impact on multiple parties*

Finally, the constant comparison method is repeated for the concepts to produce a third level of abstraction: categories. Fig. 1 gives an example of one such category.

To organize emerged categories and their mutual relationships we applied the "Six C's" coding family as described by Glaser [30]. According to Glaser this is the first general code to keep in mind when analyzing the data. Table 3 describes the terms used in Glaser's Six C's model. In our case these code categories were a good fit for the phenomena that emerged. The *Context* in our case is the occurrence of agile teams in a traditional enterprise, as discussed in Section 2.3. We identified two core *Categories*, discussed in Sections 3 (*Increased IT Landscape Complexity*) and Section 4 (*Lack of Business Involvement*). Each of these core categories arises under certain *Conditions*, and has certain *Causes* and *Consequences*. These consequences are mitigated by *Contingencies*. Finally, possible combinations of categories or parts thereof are denoted as *Covariances*. Since we have data from two companies only, we do not have enough information to reliably tell how causes and consequences vary between companies, and how mitigating tactics vary between companies. We therefore omit the discussion of the covariance part. Our adaptation of Glaser's Six C's model is depicted in Fig. 2.

The two core categories that result from our grounded theory of agile practices in traditional enterprises, based on analyzing interviews with 21 practitioners from two large enterprise organizations, are discussed in the next two sections.

3. Increased IT landscape complexity

Years of enterprise information system development typically result in a large and complex IT landscape, with an abundance of

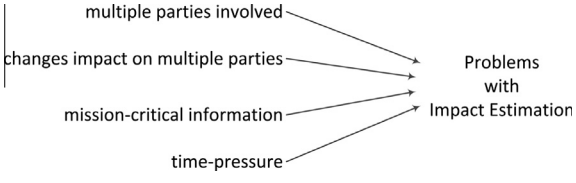

Fig. 1. Example of emergence of a category from underlying concepts.

Table 3
Six C's terminology.

Term	Description
Context	The setting where the category is at play
Condition	A factor that is a prerequisite for the category to emerge
Cause	A reason for the category to occur
Consequence	Outcomes or effects as a result of the occurrence of the category
Contingency	A moderating factor between categories and consequences
Covariance	Categories or parts thereof can co-vary with each other, meaning that a change in one category inflicts a change in the other

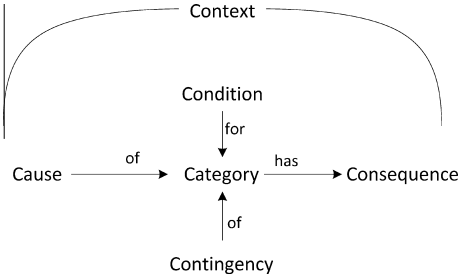

Fig. 2. Our adaptation of Glaser's Six C's Model [30].

systems interfacing in every possible – and impossible – way [77]. The level of complexity of an organization's IT landscape influences agile development teams. Our research exposes several causes and consequences of the increase in IT landscape complexity that are a consequence of the addition of agile teams. Additionally, mitigating strategies used by agile practitioners to manage IT landscape complexity are identified. Fig. 3 shows our representation of the category *Increased IT Landscape Complexity* and its constituent parts as it emerges from the data.

Especially at Company A, managing the vastness and complexity of an enterprise's IT landscape involving a combination of agile and plan-driven approaches is experienced as "the enterprise's single most threatening aspect" to the practice of agile principles. In Company B, this aspect is less threatening, as it has more of the mitigating strategies in place that are discussed in Section 3.4.

3.1. Condition: Adding agile projects to an otherwise traditional enterprise

The agile manifesto states that sponsors, developers, and users should be able to maintain a constant pace indefinitely [8]. Agile methods rely heavily on people and their skills [11], while plan-driven methods rely on pre-defined processes [11]. Changing to agile eliminates a large part of the "apparent predictability" that is provided by plan-driven methods where rigorous planning and design generate a clear picture of the IT landscape – whether that picture is valid or not is another question. Practitioners have to critically consider the application of agile practices in an enterprise environment:

> "*You can't just hack code. There are so many things in our system you have to account for, mission-critical things. Therefore, design is mandatory. With agile you have no idea what the repercussions of your development are, because you can't account for everything without thinking it through in a meticulous fashion. That's what I believe.*" —Release Manager

The pace of agile processes increases the perception of IT landscape complexity:

> "*Within your small team you closely cooperate. But we have a lot of teams and other projects or departments involved. Aligning the whole programme to get the best result, the best value for the business, is a very daunting task. Especially when we develop as fast as agile dictates.*" —Business Analyst

3.2. Causes

Participants mention the following causes of *Increased IT Landscape Complexity*: *Concurrent Development Streams*, *Separated Layer Development*, and *Different Process Approaches*. This section discusses each in detail.

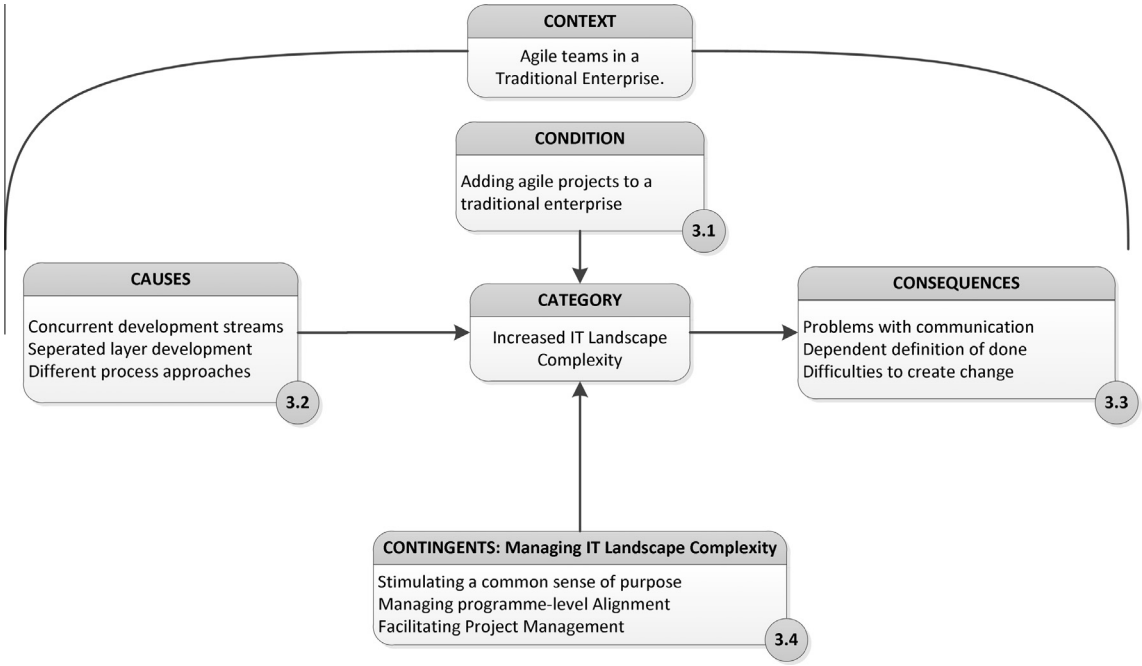

Fig. 3. The category *Increased IT Landscape Complexity*.

3.2.1. Concurrent development streams

Because of the sheer size of the projects at hand and the urgency to get things done fast – to ensure market advantage, work is divided over multiple concurrent development streams, further complicating the inter-project dependencies:

> "*The fact that we have so much streams working at the same time costs a lot of time. Aligning all interests, all functionality. Like different processes in our application, you want the user to experience uniformity in all aspects. That's hard in a plan-driven development process, but often feels like an impossible mission to accomplish when you try to develop fast using Agile.*" —Business Analyst

3.2.2. Separated layer development

A layered architecture where different parties are involved in the development at the front and back-end complicates the development process as inter-project dependencies are introduced:

> "*We want to implement agile practice as purely as possible. However, the way we implemented our architecture makes this very difficult, the existence of a front-end and a back-end and the separation of development on both ends introduces many dependencies which we have to account for in our work-items.*" —Product Owner

3.2.3. Different process approaches

Different types of systems ask for different development processes [79]. The resulting differences in process approaches applied throughout the IT landscape increases the complexity of inter-project dependencies:

> "*Those project dependencies make our work very complex, of course . . . we [front-end development] have to cope with projects [back-end] that aren't agile, but of whom we are dependent. They just employ waterfall models. That makes it very difficult to get things "Done". This introduces a lot of workarounds and eventually rework. Their design phase is long for each requirement we introduce. That makes their delivery less transparent and predictable to us.*" —Scrum Master

In addition, independent contractors are hired to compensate for the lack of specialization within the enterprise. All these contractors have their own interests that may conflict with those of the enterprise. A product owner pointed out: *"They [back-end party] outsource their work to India, I can imagine that an increase in efficiency by applying agile could cost jobs over there. That would not be in the interest of some of the people over there. So, while we want to be more efficient, contracts and their conflicting interests make some changes impossible."*

3.3. Consequences

Our research exposes several consequences of increased IT landscape complexity. The most critical ones: *Problems with Communication*, *Dependent Definition of Done*, and *Difficulties to Create Change*, are presented in the next subsections.

3.3.1. Problems with communication

Agile processes advocate face-to-face communication as the most effective and efficient method of conveying information to and within development teams [8]. Participants indicate shorter communication lines speed up their work and stimulate engagement amongst team members. However, as a result of the landscape complexity development occurs in a separated fashion, introducing the need for inter-team communication:

> "*They expect us to show initiative and communicate actively with team members and other colleagues. However, it's very difficult to communicate with people that aren't part of your team, even if they are part of the same project. Most of them aren't even in the same country or speak the same language. So face-to-face communication is totally lost in that aspect, but communication through e-mail or phone is difficult as well. On top of that you don't create a bond, a common understanding, which I think is essential during a development process. So yeah, essential functionality which depends on [back-end] software is very difficult to clarify amongst parties involved.*" —Developer

3.3.2. Dependent definition of done

The agile principles recommend the frequent delivery of working software [8]. Scrum defines *Definition of Done (DoD)* as a simple list of activities that adds verifiable or demonstrable value to the product [69]. Each workitem needs to conform to this DoD to be included in the delivery of a product increment. Dependencies with systems or product components introduced by the IT landscape's complexity impede the completion of the DoD:

> "*There is a very strong dependency with the back-end. Thankfully I sleep well, but it's very hard to conform to our Definition of Done when the connection to back-end services is part of a work-item. So you lose control over your own DoD, which makes it hard to deliver product increments ... we don't have one team, but multiple teams influence our work.*" —Product Owner

3.3.3. Difficulties to create change

The traditional project organization requires projects to be carefully documented. As a result of the document trail demanded by the project board the agile development process is preceded by a requirements phase where documents are prepared. Consequently, the agile teams are challenged by a non-negotiable scope, meaning that all requirements are typically set in stone before the agile process starts. This threatens the embrace of change:

> "*One of the things that are troublesome is the fact that things are designed at an early stage. This creates expectations for everything that follows. And we [agile team] are the final stage. So, this creates a lot of expectations if you go from project brief, to project initiation document, and finally to execution ... especially because we employ Scrum very meticulously, like purists do. We want, no, we need the freedom to have an open ending to our process ... the difficulty is that this document trail introduces fixed-price, fixed-date and a non-negotiable scope.*" —Product Owner

Agile processes harness change for the business' competitive advantage [8,23]. Participants indicate this promise as one of the main drivers to adopt agile methods. As discussed in Section 4.3.2, change is created on the fly by developers. In real life, however, teams struggle when change occurs, due to the landscape complexity. The application of different process approaches increases this complexity, especially when parties apply a waterfall model and demand requirements to be specified in early stages of the process:

> "*The back-end demands specifications of our requirements at a very early stage ... but if you want to make adjustments because you want to implement things in another, more practical way, you end up with a lot of rework.*" —Product Owner

One developer describes how changes in the front-end system were not aligned with work on the back-end system:

> "*At the start of the process, the manager of the back-end approached us and said they wanted to know exactly what we needed in October ... at that moment we weren't ready yet, so we made some assumptions. We did our best to describe in detail what our intentions were. However, at a later stage we ran into problems, because when they delivered their first code, we had had a couple of sprints already and many things had changed. Those changes were not represented in their code.*" —Developer

The possibility to create change is limited when teams are confronted with dependencies; even to a larger extent when dependent on waterfall parties.

As a result, analyzing the impact changes have on the programme is mandatory. Impact on dependent projects can manifest itself in changes in work-items, priority changes, or delayed feature delivery:

> "*They [Business Stakeholders] expect that every change they introduce can be picked up unconditionally. While this is true in most cases, you have to estimate the impact it has on the whole process. Furthermore, you cannot just change something that has already been built. For example, the department that generates content for the pages introduces new pages in the middle of our sprint and expects them to be picked up immediately, with total disregard of our baseline. This forces us to implement these pages in the next sprint ... Business needs to take into account the context. It is not just one website which has to be realized. A typical change has repercussions on many levels. Within our process, but outside it as well. We can't 'hack' in every change. We need to take into regard the stakes of all parties involved.*" —Scrum Master

3.4. Contingencies: Managing IT landscape complexity

Several strategies are used by the teams to manage the landscape complexity. Some of the strategies were mentioned as possible solutions by participants of the first company and then later confirmed by participants of the second company. Strategies to manage this complexity that emerged from the data include *Stimulating a Common Sense of Purpose*, *Managing Programme-level Alignment* and *Facilitating Project Management*. These are discussed in the next subsections. Interestingly, the first two strategies mainly emerge from the second company studied, where participants experience the challenges mentioned above to a much lesser extent than participants from the first company do. From this we conclude these strategies help mitigate the consequences of *Increased IT Landscape Complexity*. Positive as well as negative examples of *Facilitating Project Management* are observed at both companies.

3.4.1. Stimulating a common sense of purpose

Section 3.3.1 discusses how different process approaches poses threats to communication. By getting to know each other, one stimulates a common sense of purpose, and barriers between interdependent parties can be reduced:

> "*The project is very broad. A lot of parties are involved ... the difference in process approaches introduces frictions between parties ... but over time you get used to each other and they [back-end party] got more lenient with us, more cooperative. Slowly you grow towards each other and you start to accept things from each other. That's very important, you have to cooperate to make this project successful.*" —Business Analyst

As a result, the creation of and reaction to change is improved, and the value delivered by the programme is increased:

> "*Our short communication lines with [back-end party] allow us to cooperate effectively. It ensures that we can act quickly on changes at either end. We all need to be flexible, otherwise things will not work out. Along the way a common understanding has developed. You try to work together instead of being competitors ... One should set aside one's own interests, and help increase the value to the entire organization.*" —Scrum Master

3.4.2. Managing programme-level alignment

Aligning knowledge and decisions at a programme level proves to be essential to ensure sufficient reflection of project and team interests within the teams, while increasing the value of the developed product. Managing Programme-level consists of *Combining*

Product Backlogs and *End-to-End Representation in Team*. Each of these strategies is discussed in the next paragraphs.

3.4.2.1. Combining product backlogs. It is easy to lose track of priorities within a complex enterprise environment where the development of different parts of a system as a whole is dispersed over multiple teams and/or concurrent projects. Combining product backlogs of teams that depend on one another helps teams plan and align dependent work items:

> "*Now, we have combined our product backlog with multiple teams, one of which is [back-end project]. Work-items of all parties are ordered by priority. During that process we agreed with [the back-end project] that when one item of ours has a high priority, the item (containing the used business services) of theirs that is related should be high too. This helps a lot with planning and alignment.*" —Scrum Master

Tooling can help in this respect. Setting up a project management environment that is available to all involved parties, where the Product Backlog is maintained in one single place, helps to keep track of any changes made by concurrently working teams:

> "*Within our [Back-End] project we use [Tooling] to keep track of our Product Backlog. Our three teams use this tool to keep track of the priorities on the Master Backlog while working on their respective Sprint Backlogs. A team member from team A can exactly see who is working on what workitem. This helps u a lot to keep track of our priorities.*" —Release Manager

3.4.2.2. End-to-end representation in team. End-to-end representation means that all knowledge required by the team is available within the team. Face-to-face communication is an important pillar of the agile principles. When an agile team has dependencies with other teams or projects – back-end parties in particular – face-to-face communication is impossible. Developers from back-end system projects are added to the agile team to assist when dependencies apply to the current Sprint:

> "*When our items depend on the back-end, whose team is typically not employing an agile process, we add a back-end developer to our team to implement the changes we ask for.*" —Scrum Master

> "*When we are dependent on external parties, we always ask for one of their developers to work alongside our team. At least for a few days or a week. To check if things are implemented efficiently. He can pay attention to the dependencies we have in the Sprint at that particular moment. So, we try to involve them as much as possible in our team.*" —Scrum Master

The back-end developer helps the front-end team by sharing his knowledge. He also provides short-term solutions to problems, ensuring the teams progress and flow:

> "*[Back-end Developer] has the responsibility to add realism to the front-end regarding the proposed solutions. [Back-end Developer] knowledge of the back-end catalogue, where our services are described, enables [Back-end Developer] to make estimates for these solutions and advise on the actions to be taken. [Back-end Developer] is able to update services immediately when applicable, or to provide stubs – which we can do in a matter of hours now. [Back-end Developer] is also able to make estimates, which helps the team with their Sprint Planning.*" —Release Manager

3.4.3. Facilitating project management

When adopting agile methods within an environment where a traditional project organization exists, one of the first persons that feels change is the project manager. Views expressed by participants on his role vary between *'his input is essential to our process'* and *'he helps us a lot'* to *'I always found his role to be quite odd in our scenario'* and *'our project manager is kind of a misfit, if you ask me'*. Whatever the stance taken by the participants, throughout the interviews the project manager was mentioned as playing an integral role within the daily activities of people.

The adoption of Scrum introduces two new roles – the Product Owner and Scrum Master – that influence the set of responsibilities of the Project Manager. Due to the shift in context, the project manager's role changes from his traditional directive attitude to a more facilitating one.

The project manager's "new" role manifests itself in different ways:

3.4.3.1. Facilitates alignment. An enterprise organization typically employs plan-driven processes. The fast and lean way of working agile conflicts with this way of working. In our context there are remains of a project organization which demands a well-documented process throughout. The success of an agile initiative within this environment is greatly influenced by the Project Manager and how he approaches his role in the process:

> "*For one project we had a Project Manager who really took on a facilitating role between all parties involved; and that project went very well. Super. The Project Manager had a very big role in that success. The only point of critique – speaking as an agile purist – was the absence of any business involvement during the project, but the Project Manager did very well in that respect. He was a great governor. He governed all things with external parties we were dependent on.*" —Scrum Master

The Project Manager coordinates all that surrounds the self-organizing agile teams. He makes sure that all dependencies, whether at business-level, programme-level, or from suppliers are met at the right moment in time:

> "*We still have a Project Manager, we need one still as well . . . think of legal issues, or security issues that need to be taken into consideration. Or when you have multiple teams working on one project, in that case the Project Manager is important to align these teams. Ensuring that the ends meet at the right time. So, yeah you still need the Project Manager, but his role has become much more about coordinating the process instead of directing it.*" —Scrum Master

His main means of succeeding in aligning all dependencies is communication. He informs his teams of the dependencies at hand and the facets of their activities these dependencies influence.

The Scrum team is, in principle, a self-organizing unit. Their formal task is to deliver a potentially releasable product increment at the end of each sprint [69]. It is a daunting task in itself to turn sprint backlog items into the next product increment and requires focussed developers to succeed. As a result, the focus of the Scrum team is inward, to their own priorities and their own success.

The complexity of an enterprise environment introduces multiple dependencies with other projects, both between concurrent development projects and between the front end and back end of the system. The IT programme has to take into account all these dependencies to ensure different projects do not end up being misaligned. The project manager monitors the bigger picture and keeps the team informed about the repercussions that some backlog items have on the environment of the team.

> "*What you have to keep in mind is the product backlog. The sprint backlog is filled based on this product backlog, which is created based on priorities. But there are, of course, bigger things at play . . . therefore, I have my interests too, for what ends up on the backlogs. Things that need to be delivered in time. So the Product Owner and the Scrum Master make most decisions, but I have to be able to*

inform and contribute to this process, like: 'It's nice you want all of this, but these things have to be added as well, otherwise we won't be able to deploy anything.' Like that." —Project Manager

By collecting information from his environment and by informing the team and participating in their decision processes, the Project Manager is able to ensure alignment at the programme-level.

"*To ensure alignment within the environment, I look around. I watch other teams and projects, check where dependencies are, how they develop. I keep an eye to the applicability for the group as a whole, check where the biggest risks are. If we have to mitigate them first or last. Of course I use the input of my environment, for example my Business Analysts can advise me on complexity issues, but developers help me as well.*" —Project Manager

3.4.3.2. Well-embedded in the organization. Agile initiatives cannot grow in isolation [38,39]. Without the support of the IT and business organizations, establishment and growth of the agile initiative is impossible. This feeds the need for someone to actively involve the organization in the agile process by generating awareness and understanding. A typical Project Manager from an existing Project Management pool within an enterprise organization is well connected:

"*[Project Manager] is of use in political games as well. Generally our Project Managers are well embedded in the organization, so they are able to arrange things much faster and often remove noise within the higher regions of management ... on many occasions they are able to let people walk for them ..., I won't get the Marketing Director to do things for me, but the Project Manager can – maybe with some help from the Programme Management, but you get the point.*" —Product Owner

The level at which the Project Manager is embedded within the organization also enables him to escalate when situations call for it. Furthermore, he is able to defend his team against higher management without risking his own head:

"*[Project Manager] is able to escalate within the back-end parties as well. For example, [Back-end Party] was supposed to deliver on the 15th; if they didn't, we weren't able to guarantee that we would succeed to finish our sprint and give a demo. Well, the 15th arrived and [Back-end Party] wasn't able to deliver. At that point, [Project Manager] said: 'If we do not get that stuff before the 30th we will not be able to deliver anything at all!' So that day just a part was delivered to us and we had to work really hard to get things done. But, with [Project Manager] escalating to higher management, at least some stuff was delivered. On top of that, he also has the right amount of influence in higher management to be able to say, that we won't be able to finish things – without the risk they chop of his head.*" —Developer

3.4.3.3. Involved in team process. Members of the team build a common sense of responsibility and that brings them close together. The Project Manager can increase his influence on the team by staying closely involved with their process:

"*Because I personally appropriated the role of constantly checking things, I end up being around the team much more. I used to sit at my desk a lot more in the old days. But now I have to spend much more time on the floor. Most of my people are in the same room with me, so that helps a lot. So there is eye contact at the very least, when I'm around. Communication got a lot easier since we sit together ... the way things are going now, contact is great!*" —Project Manager

However, when a Project Manager neglects his team and does not invest time to show interest, he risks to be left out of the process:

"*There are other teams where the Project Manager has a clear presence, but ours is just not that involved. He only attends our standup from time to time, but never participates. And after the meeting he usually leaves as soon as possible ... he just does not seem to be engaged with our cause ... as a result, he ends up being the odd man out.*" —Product Owner

3.4.3.4. Relationship with the product owner. A major difference between an agile team and a traditional team is the fact that decisions about the contents of the product are made by the Product Owner, not the Project Manager. The Project Manager focusses on the 'how' of a project, the Product Owner focusses on the 'what' of a project.

"*[Product Owner] decides what to develop. [Product Owner] does not decide on how things are to be developed. [Product Owner] has a big say in the priorities of course. What is important to [Product Owner] to develop. Of course we choose what is important to develop.*" —Project Manager

The Project Manager assists the Product Owner by managing the alignment of all involved parties other than the business stakeholders. Making sure that the Product Owner can focus on the realization of a product that adds business value.

"*The way I see it, we still execute our projects in the existing project structure; hence, there are a lot of activities that belong to a Project Manager's responsibility ... I want to focus on the realization of a product ... I don't have to worry about informing or aligning other parties besides the business stakeholders.*" —Product Owner

3.4.3.5. Relationship with the scrum master. A traditional project manager would directly interact with the team of developers in case he has a problem that needs to be solved. An agile team has its own autonomy during a sprint, and a project manager is not supposed to interfere. The Project Manager in this case has to take care how and when approaching his team. When some event takes place that he perceives to be serious enough to direct his team's attention to, he risks productivity loss and eventually the success of the sprint:

"*Incidentally I certainly negatively influence the team process, from a Scrum perspective ... if I run into problems during testing, or someone notifies me of such a problem, I tend to go straight to the developer responsible for that part. But I immediately influence the Sprint Backlog and productivity, because I order them to do stuff that is not on the [Sprint Backlog].*" —Project Manager

The Scrum Master helps the Project Manager in this respect by acting as a buffer between the Project Manager and the team, ensuring productivity and Sprint success:

"*When you don't spend enough time with your team, some of them lose focus ... they miss you at some conversations. [Project Manager] from time to time tries to order the team around when he has the chance. That does not work, of course. At that point, they need me around to kick him out and ensure focus on the Sprint Backlog. Otherwise, [Project Manager] endangers our productivity and the feasibility of our Sprint Backlog.*" —Scrum Master

3.4.3.6. Relationship with the development team. When involved with the process the Project Manager builds a close bond with the Development Team. Through agile practices, the communication between developers and the Project Manager becomes much more informal. This can have disadvantages as well:

"*Sometimes you are tinkering with some stuff that is not really on the Sprint Backlog, just for the fun of it. And you create stuff that shouldn't be created at all yet. When [Project Manager] watches over your shoulder and sees something slide really neat, he will get the feeling that it's really easy to realize. You have to be really discrete, otherwise he will underestimate the work to be done. That's difficult in an agile team. Everybody sees what everybody is doing.*" —Developer

The Project Manager communicates with the outside world. Since he is so close to the Development Team, he knows the details of what is going on and may be inclined to, for instance, promise certain functionality that is not yet ready to be delivered. The Development Team members have to pick their words carefully to not reassure the Project Manager of things that are not yet that certain:

"*To not make him escalate too often we sometimes have to watch what we say when [Project Manager] is around. I tend to show things I like, but I have to be careful. For example, the services we have to integrate are dripping in at the moment and some seem easier than others, so I say: 'With a little overtime we can get this done before Friday!' That's my gut-feeling at that moment. As a result, [Project Manager] feels reassured, which he communicates in another meeting. Well, if we don't get it done by Friday, we have a problem.*" —Developer

3.5. Wrap up

The IT landscape complexity increases when agile teams are added to a plan-driven development organization. The causes of this increased landscape complexity are: (1) work is divided in concurrent development streams, (2) parties within both the traditional IT department and the agile projects are involved in development, and (3) different process approaches are involved in the development. The consequences hereof are: (1) communication between parties involved gets more complicated, (2) dependencies between the agile project and other projects makes it difficult to decide when an increment is ready, and (3) the possibilities to create change are limited.

The increased landscape complexity can be dealt with successfully by:

- creating a common sense of purpose, which has been shown to be a major motivator [66];
- managing alignment at the programme level, by combining product backlogs and ensuring representation of both the back-end enterprise development and the front end agile projects in each team;
- facilitating the role of the project manager which, in this new context, shifts from a traditional directive attitude to a more facilitating one.

Each of these mitigating actions improves communication between parties involved and thereby ease the management of dependencies and the possibilities to create change.

4. Lack of business involvement

Fig. 4 shows our representation of the category *Lack of Business Involvement* and its constituents as it emerges from the data.

4.1. Condition: Business is focused on the traditional development paradigm

One of the agile principles states that business representatives and developers must work together daily throughout the project [8]. As opposed to a plan-driven project, agile projects demand more input and continuous involvement of business representatives. The responsibilities of business people are extended with: (1) helping the construction of user stories, (2) discussing and prioritizing product features, and (3) providing feedback to the development team in a timely manner [23]:

"*I can understand the attitudes of my colleagues . . . we are used to the paradigm of making up requirements and then throwing them*

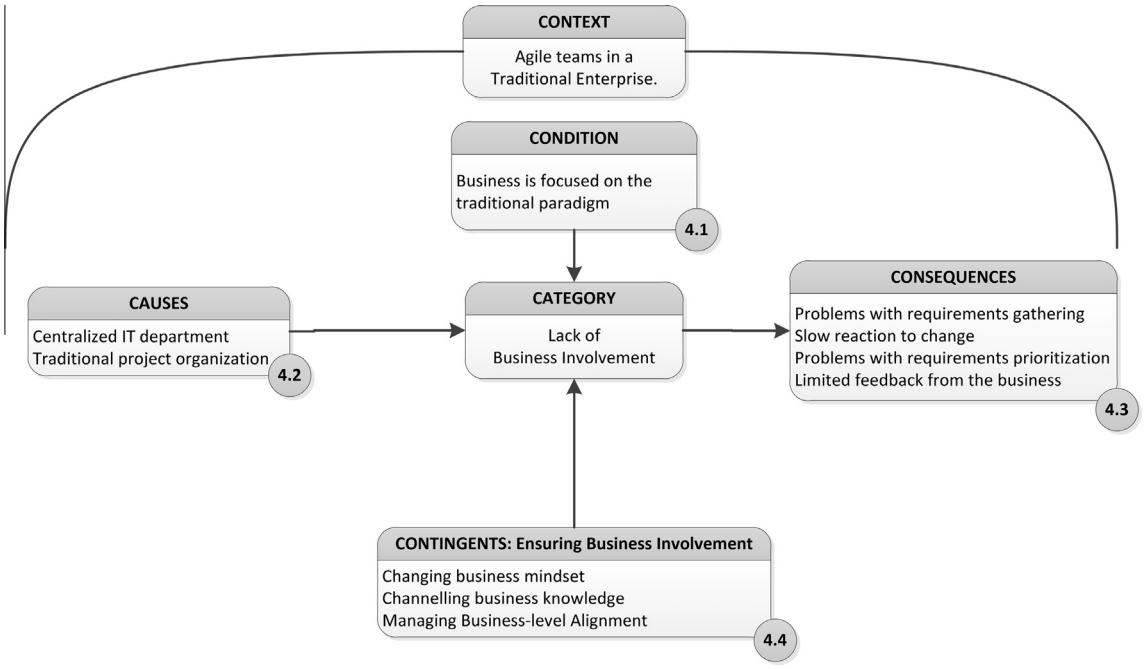

Fig. 4. The category *Lack of Business Involvement*.

over the wall, showing up months later to check if everything is done. That's how we've always done it. We're not used to being constantly contacted about development and other things at IT." —Business Stakeholder

Most participants indicate that the business is not as involved as agile methods demand:

"*Not everyone attends our sprint demonstrations. Some people of the business visit our site once in a while creating a bond. Others do not feel that obligation, and expect us to organize meetings specially for them ...*" —Scrum Master

"*Most of the business representatives do not care enough to be constantly involved. They often say: 'come on, I gave you my requirements just build the damn thing.' As a result, they often completely disregard our requests when something changes.*" —Product Owner

Ensuring business involvement is perceived as "one of the most challenging parts of the agile adoption process."

4.2. Causes

Two causes that lead to a lack of business involvement are indicated by participants: *Centralized IT Department*, and *Traditional Project Organization*. The following subsections describe these causes in detail.

4.2.1. Centralized IT department

Centralized IT departments slow down the delivery of IT services and create a gap between business representatives and IT professionals. The centralized structure of the IT department is perceived to influence the involvement negatively as budget responsibilities are not with the Business Stakeholders, but with IT management. As a result, senior IT management interferes in the relation between development team and business, as they want to maintain full control over the process. Moreover, business shows a lack of urgency, as they do not feel responsible when the project fails:

"*Our IT department is centralized, which means that budget responsibilities lie with our senior [IT] management. As a result our management wants to keep control and consequently they assign the Product Owner role to IT employees rather than Business Stakeholders. Additionally, Business Stakeholders clearly feel less urgency for their projects as they are not part of our team and do not feel the pain when the project fails, it's not their money after all.*" —Scrum Master

4.2.2. Traditional project organization

Traditional roles still exist due to the project organization. These roles demand prerequisites for a project to start, and ask for extensive documentation. To cater for that, an introductory phase – where documents, including a marketing brief, a project brief and a project initiation document are prepared – is conducted before actual development starts. This gives business people the impression that their work is done after the project initiation document is delivered. Consequently their involvement is negatively influenced:

"*We have a waterfall period before our agile development. Simply because some people have old function descriptions. As a result they need paperwork to govern their activities. They need that marketing brief, project brief and the project initiation document to report back to the IT organization. This introduces a workload of three months before the real development commences.*" — Scrum master

After all, business is used to work with plan-driven development [11]:

"*I can understand the attitudes of my colleagues ... we are used to the paradigm of making up requirements and then throwing them over the wall, showing up months later to check if everything is done. That's how we've always done it.*" —Business Stakeholder

4.3. Consequences

Our research exposes several consequences of the lack of business involvement. The following subsections give an overview of the most critical consequences perceived by participants: *Problems with Requirements Gathering*, *Slow Reaction to Change*, *Problems with Requirements Prioritization*, and *Limited Feedback from the Business*.

4.3.1. Problems with requirements gathering

At every iteration, user stories – representing requirements – are to be provided by business representatives. Additionally, business representatives are responsible for the clarification of business requirements. In real-life, agile initiatives face challenges getting requirements from the business stakeholders, resulting in unnecessary delays and loss of productivity:

"*On some occasions you have to collect requirements in a very short amount of time. The rigidity and vastness of the enterprise in particular poses challenges during this process. Agile is fast and furious. This makes it very difficult as several parties are unable to keep up with this pace. As a result, I cannot get requirements on time or when I do get them, they are incomplete or unclear. Additionally, our landscape consists of many different systems and projects, you have to consider all dependencies in that short window of time as well.*" —Product Owner

Consequently, the agile process is often preceded by traditional requirements elicitation and analysis phases (see Section 4.2.2) and extensive lists of requirements are provided. Nevertheless, further clarification is needed later in the process. At that point, however, the business is much less involved as they think their job is done:

"*The problem with business stakeholders is the fact that they are still used to old habits. They feel that their job is done when the marketing brief and project brief are finished and all requirements are elicited. It is really a case of we-them mentality. They throw their lists of requirements over the fence and expect that their responsibility and involvement stops there. At the deadline they wave the lists in our faces when stuff is not done.*" —Project Manager

When requirements are unclear and the business representatives are not available, teams are forced to make assumptions about the business' requirements:

"*In some projects we had to make assumptions about certain requirements, for the simple reason that context about them is missing and the business stakeholder is not or at least not in a timely manner available to us.*" —Product Owner

4.3.2. Slow reaction to change

One of the most important facets of agile methods is the ability to create change [23]. The agile manifesto boldly claims that the best architectures, requirements and design emerge from self-organizing teams [8]. As a developer points out *"As developer you constantly run into flaws in the concepts that are described in the user stories, simply by developing, testing and experiencing the resulting product. At that point we have, and use, our freedom to implement*

a more practical solution." User stories of an agile project are constantly changing. When changes occur the business representatives are responsible to reassess the user stories. However, agile teams faced challenges in receiving reactions on changed requirements from the business:

> "*The business still employs traditional thinking. They are used to thinking when we cannot finish something today we just finish it next week. Parkinson's law . . . When, for example, we need to finish a report on some sort of change, the issue of reviewing arises. 'If someone cannot review it until next month; well, then we have to wait. Only when the report is reviewed, can we estimate when we are done based on the findings of the reviewer.' It is difficult to introduce this culture change and get things done faster.*" —Scrum Master

When unable to react to changes in a timely manner the business can cause unnecessary delays and loss of productivity:

> "*One of the user stories I am responsible for is pending for feedback of the business. It has been 67 days now. We have taken it off the sprint backlog now, but it took a couple of days before we realized we would have to wait too long for reactions.*" —Developer

Similarly, some teams experience a total disregard of reaction to change by the business:

> "*Most of the business representatives do not care enough to be constantly involved. They often say: 'come on, I gave you my requirements, just build the damn thing.' As a result, they often completely disregard our requests when something changes.*" —Product Owner

Without reassessment of changed requirements, teams are forced to make assumptions about the business' needs and priorities, or to ignore the change altogether. Due to these inaccurate assumptions or ignorance of change, the team builds features that are not as the business requires. As a result, the team has to perform rework which incurs additional costs for the business:

> "*I have experienced projects where the total disregard of involvement results into a lot of rework. Because we build things they do not want at all . . . our budget explodes due to that rework.*" —Project Manager

The traditional culture of business departments conflicts with the fast agile processes. Business representatives are not aware of increments and feel they have enough time to react to requests from the agile team. They are less involved or not involved at all. As a result they cause delay or loss of productivity, forcing the teams to make assumptions about business needs. Eventually, this introduces rework when the gap between needs and delivered value is too big.

4.3.3. Problems with requirements prioritization

Agile methodologies require business representatives to order the user stories by priority. This ordering indicates which user stories are to be completed first and is driven by their business value. The process of prioritization doesn't always come naturally to business representatives new to agile:

> "*We are used to just make a big list of requirements and toss it over the fence – so to speak – and to come back after a few months to evaluate what has been done and what not. Everything we put on this requirement list is important to us. So we expect everything is done. So the process of prioritizing requirements seems redundant from our point of view. Everything needs to be in there.*" —Business Stakeholder

Furthermore, the process of prioritization within enterprise projects is perceived as a tedious and daunting task, as a lot different interests are at play:

> "*Very difficult to us is the process of assigning value to our work-items. How do you decide this value? That process has a steep learning curve, and you can only learn by experience . . . some value assignments are pure common sense, but when more intangible facts come to play like customer satisfaction versus profitable aspects. You tell me, what is more important. That is a very difficult consideration.*" —Product Owner

The difficulties in getting prioritized requirements from the business can make it very hard for the teams to get a grasp of what features to develop and deliver first, introducing a threat to the business value generated. In contrast, when requirements are adequately prioritized by the business, the business value delivered in each sprint is maximized.

4.3.4. Limited feedback from the business

Customer feedback is an invaluable tool to ensure that the product developed and delivered incrementally adds value to the business. It requires the business representatives to provide feedback on developed features. Most teams face challenges getting the important business stakeholders to attend to sprint demonstrations:

> "*Not everyone attends our sprint demonstrations. Some people of the business visit our site once in a while creating a bond. Others do not feel that obligation, and expect us to organize meetings specially for them. We simply do not have the time to do this for each business person.*" —Scrum Master

> "*Most business stakeholders have busy agendas. As a result, the priority of their projects at IT is low. So they do not see the value of coming to our sprint demonstrations enough, making it difficult for us to be sure we are developing exactly what they want.*" —Project Manager

When business feedback is insufficient, teams are not able to assess how the features meet the requirements. In contrast, when business involvement is sufficient teams have better and more frequent communication with important business stakeholders, increasing the value delivered.

4.4. Contingencies: Ensuring business involvement

A couple of interesting strategies are used by practitioners to ensure business involvement by mediating between the aforementioned causes and consequences. These strategies include *Changing Business' Mindset*, *Channelling Business Knowledge*, and *Managing Business-level Alignment*.

4.4.1. Changing business' mindset

As an operational manager pointed out *"Enterprise organizations typically have an older employee base. These employees are set in their ways . . . are certainly not used to the fast way of working . . . and considering IT they just feel that they have nothing to do with it after they vented their requirements"* and, as a result, business stakeholders have to be thoroughly convinced of the added value of agile before they commit extra time to the project. However, this major change in culture requires time:

> "*Agile is not only about a change in process, but also a change in culture. This is something they [business representatives] have to experience for themselves and that requires time.*" —Scrum Master

Generating awareness and understanding of agile processes is essential to get business stakeholders to change their mindset and get involved in the process:

"*I'm part of the Agile Workgroup. Our task is to roll out the concept of agile throughout the whole organization. In this stage this means creating awareness and understanding amongst all ranks in the business and IT. I frequently present Scrum master classes to a delegation of certain business departments ... in these classes we communicate the benefits of agile and present some recent success stories from within our own organization ... more and more people are getting exited by the concept, which makes our task easier as they are getting involved!*" —Scrum Master

The feeling of being in control of development is an important aspect that aids to the change of mindset:

"*If done right. If a business stakeholder is really involved in the process you really see him get more exited every sprint. They really feel in control. They can direct the way their product is developed.*" — Scrum Master

"*I really enjoy the control I have over the project. After every sprint a demonstration is given and I have the opportunity to assess the value of the product and direct where necessary!*" —Business Stakeholder

Finally, higher management has a very important role in changing the mindset. When the CIO of the company is a strong proponent of agile methods, things seem to go a lot smoother:

"*Our CIO has been replaced recently. The first thing he said was that he wanted to extend the agile initiative. From that moment on our initiative began to gain momentum and the awareness within the organization grows exponentially at the moment!*" — Scrum Master

4.4.2. Channelling business knowledge

Enterprise environments are characterized by a vast amount of departments and business units [77]. The same is true of the organizations we study. As a result, the ownership of an enterprise information system, or parts thereof, typically belongs to many different business stakeholders. Scrum dictates that a team has but one Product Owner; who has mandate for the product. In practice this distributed business knowledge should be channelled through this Product Owner, making him the representative of all business stakeholders involved:

"*Knowledge is highly distributed within the business. Scrum defines for a team to have only one Product Owner. As a solution, the role is assigned to an employee of the [particular] department; channelling all of the business knowledge through him/her. The Product Owner is situated at our location. Which is perfect for communication.*" —Scrum Master

"*Our Product Owners represent the business, but they are not really the owners of the product. Their function is to keep things maintainable, from a Scrum perspective you have to have a team member that has mandate ... That is good for our project, otherwise we would have to involve all those business parties into our project. That is channelled now; and that is very convenient.*" —Project Manager

4.4.3. Managing business-level alignment

Aligning knowledge and requirements at a business-level proved critical to ensure sufficient reflection of the business demands in the product developed. Managing business-level alignment is reached by Intensive Stakeholder Communication.

4.4.3.1. Stakeholder communication.

When business knowledge is channelled through one representative, the actual stakeholders – or owners – of the product are not as closely involved as one would wish:

"*While [Channelling Business Knowledge] is a solution for the distributed knowledge, it results in the fact that business is not closely involved in our process. Some people of the business visit our site once in a while creating a bond. Others do not feel that obligation, and expect us to organize meetings specially for them. We simply do not have the time to do this for each business person.*" — Scrum Master

The person channelling business knowledge has a great responsibility to reflect the business stakeholders as much as possible in the process. Communication is very important tool to succeed in this aspect. Multiple communication strategies emerged, including:

- Collaborative Demo's: Demonstrating end-to-end functionality in collaborative demo's makes the business more aware of the dependencies at play within the system. This makes it easier for the Product Owner to explain situations and their consequences to the business parties involved, helping them to make well-founded choices on functionality.

"*The business has to understand why certain requirement cannot be met. An important role for me is to help them see this. You cannot just say: 'No, that cannot be done.' The most important thing for the business is to realize themselves why some solutions are not feasible. Transparency offered by Scrum aids in that aspect, but also our collaborative demo's. We are able to present a full picture to the business.*" —Product Owner

- Face-to-face communication:

"*I am a proponent of one-on-one contact, almost informal. Now, for example, I take some examples and get a cup of coffee with various stakeholders to align certain things. Based on the Product Backlog and the priorities I have come up with of course. I want to be certain that my decisions are widely supported by my business stakeholders.*" —Product Owner

- Frequent stakeholder sessions:

"*Once a week we organize a session with all the business parties involved. In this session we evaluate the progress and discuss situations. Furthermore, we present possible solutions to choose from. All this to align their wishes.*" —Product Owner

- Intermediate demonstrations:

"*Yes, we agreed the stakeholders are to be informed during the Sprint Demo, but I feel that this was sometimes to late. We ended up discussing a single screen for one and a half our for example. That is not efficient, obviously. Everybody meddles with the discussion, even if it has nothing to do with their department ... So I individually demonstrate things to certain stakeholders as well, just to ensure that their stuff is right and we do not have to discuss it in public.*" —Product Owner

4.5. Wrap up

Business involvement is difficult to realize in an environment used to a more traditional mode of software development. The

causes hereof are: (1) the IT department is still centralized, which creates a gap between business and IT, and (2) projects are still organized in a plan-driven way, with a startup phase in which most of the involvement of business representatives is concentrated. The consequences hereof are: (1) problems in getting the requirements from stakeholders, (2) slow reaction to changes from the business, (3) problems with prioritizing requirements by the business, and (4) limited feedback from the business on features implemented.

Business involvement in this context is improved by:

- Changing the mindset of business stakeholders; if business stakeholders become more aware of and better understand how agile process work, it becmoes easier to gather and prioritize requirements, and get more and earlier feedback from the business.
- Channeling business knowledge through the Product Owner, so that there is one business representative to deal with.
- Aligning knowledge and requirements at the business level, e.g. through collaborative demo's and stakeholder workshops, to realize frequent reflection of the business on features implemented.

5. Discussion

In this section we relate our findings from Sections 3 and 4 to research reported by others.

Agile processes champion emergent architecture, requirements, and designs [8]. Our research shows the emergent character of these project aspects becomes more difficult to manage with each dependency introduced by the environment. Enterprise systems are an important phenomenon in the organizational use of information technology and have been widely characterized as complex, and the implications of any change are hard to oversee [77]. Moreover, enterprise systems typically contain mission-critical information and processes, resulting in an emphasis on security and legal issues during development [77].

Several articles study the compatibility/incompatibility of agile and plan-driven methods using CVM, the Competing Values Model [25]. CVM distinguishes two dimensions: change vs. stability, and internal focus vs. external focus. Based on these dimensions, four types of organization cultures are distinguished: group culture (change and internal focus), development focus (change and external focus), rational culture (stability and external focus), and hierarchical culture (stability and internal focus). Plan-driven development is usually associated with the hierarchical culture, focussing on efficiency, productivity, and the like. Agile methods on the other hand best fit the group culture, with its emphasis on flexibility and interpersonal relations [41]. Other studies [14,18,80] use similar categorizations to discuss the differences between plan-driven and agile methods.

Agile software development has been considered a misfit for large enterprise information systems development by many for a long time [77]. However, a rapid changing market forces organizations to more flexible software development [19], hence the shift to agile methods in many an organization. The relation between plan-driven and agile methods is investigated in a number of case studies [68,44,70,78,74,37]. These case studies generally conclude a fit is possible. Some suggest to try to change the culture of the organization, others suggest one needs to adapt the agile practices to ease interaction between the cultures. In the organizations we studied, both occurred.

Lindvall et al. [51] found that individual projects in a large organization often depend heavily on their environment in several ways. First, work is often distributed across several teams, introducing communication and coordinating issues between the teams. Second, teams have to fit into the standard processes and quality systems defined by the organization. Finally, cultural differences between teams and their environment complicate interactions between them; they noticed that agile teams experience challenges when interfacing with traditional development teams. Lindvall et al. [51] suggest tailoring agile methods to the environments is an absolute necessity for agile processes to succeed within large organizations. An observation supported by many [1,14,23,24,46,45]. We endorse this view; in contrast to opinions of some 'evangelists' [42,75], enterprise factors force agile teams to consider the context applicable to their situation. Kurapati et al. [47] also found that many practitioners do not apply agile methods "out of the box", but rather select the practices that fit their situation. Such adaptations also increase the perceived compatibility between the approaches and hence help overcome acceptance challenges of agile methods [18]. However, as a product owner at Company A pointed out: *"the less rigorous you implement agility and the more you adjust to the existing environment, the more you risk not making that essential culture shift. You end up doing some of both worlds and lose some essential benefits you aim to get from agile development,"* One should be extremely careful not to change the processes too much to adhere to the environment. For example, Company B decided to assign Scrum's product owner role to an IT representative, rather than a business representative. As a result, the feel of business for agility and, by extension, business involvement decreased drastically.

According to Baskerville et al. [7] a major shift took place in just two years (from 2001 to 2003). *"The changing market and IT economy were having visible impact on firms that were visited."*. While companies operated with *"significantly lower capital"* than before, they were *"still under pressure to deliver software at high-speed"* [7, p. 549]. They found that balancing speed and quality became invaluable to the development process. Especially focussing their practices on prioritization and estimation of projects and functionality [7, p. 551]. An essential consequence of *Increased IT Landscape Complexity*, as presented in our study, is *Difficulties to Create Change*. Our study confirms the experiences observed by Baskerville: the importance of change grows with the acceleration of development. However, the combination of fast-paced development and landscape complexity impedes the creation of change.

In their fourth study in 2008, Baskerville examined three companies that had implemented Scrum. Common to all three companies was that they were applying Scrum for some parts of their software development process and a more traditional approach for other parts of their development efforts [7, p. 551]; contexts similar to those of our study. In their article they categorize certain Scrum artefacts as *Boundary Objects* and the project manager as a *Boundary Spanner*, both categories adapted from articles by Star and Griesemer [71] and Levina and Vaast [50]. A boundary object is an object that mediates between two communities, in this case between the Scrum team and the customer organization; prioritized user stories are an example. A boundary spanner likewise is a person that mediates between the two communities. In [73], Strode et al. discuss the role of boundary spanning activities and boundary spanning artefacts (i.e. objects) in the coordination in agile projects. A boundary spanning activity occurs when different groups (separated by location, hierarchy, or function) in an organization interact to share expertise [50]. The activities discussed in Section 4.4.3 can be understood as boundary spanning activities. Additionally, the boundary spanning role of the project manager is identified as well.

The additional need for the Project Manager to change from a directive to a more facilitating attitude to fulfil his role effectively is acknowledged in many studies. Cockburn and Highsmith [20] succinctly describes the difference as *"agile companies practice leadership-collaboration rather than command-control management"*. Augustina et al. [6] and Vinekar et al. [81] arrive at similar

conclusions; they found that agile teams ask for managers that are adaptive and collaborative, managers that act as facilitator and coach. A dissenting tale is told by McAvoy and Butler [54], who observe that agile teams may suffer from groupthink or the Abilene paradox. In groupthink [43], the drive for consensus tends to suppress disagreement and critique. The Abilene paradox is somewhat similar to groupthink: one person makes a proposal, which is then followed by another person, but eventually results in disappointment for both. The first person intends his proposal as an open suggestion, and the second person does not want to be impolite, and consents. But in fact, both persons do not like the idea [35]. Both these processes may occur in agile projects, where decision making is consensus-based. McAvoy and Butler [54] suggest the project manager in an agile project should also play the role of a devil's advocate.

Agile development teams are influenced by the level of business involvement in their projects. Insufficient business involvement has negative implications on the agile teams, while sufficient business involvement has positive implications on the teams.

On the whole, our results show that business involvement is critical for agile practices to be performed effectively. This is consistent with earlier studies[1] [11,34,38].

The introduction of agile methods into an otherwise plan-driven organization is often perceived as complex [20,28,76] and raises challenges [17,27,61,55,62,65]. Cockburn and Highsmith [20] observes that *"an agile team working within a rigid organization has as difficult a time as agile individuals working within a rigid team"*. Svensson and Höst [76] describe the introduction of agile methods in a maintenance organization, and found the need to redesign many of the practices in order to fit their particular environment. Petersen and Wohlin [62] identified several challenges that are also encountered in the organizations we studied: the need for extensive coordination between teams, requirements engineering taking a long time, and long waiting times because agile teams wait for the business to take decisions. Cao et al. [17] identify three types of challenges when adapting agile methods to fit a particular context:

- Development process related challenges, such as the upfront requirements engineering process in plan-driven method versus the emergent requirements in agile processes.
- Developer related challenges, such as the collocation of team members, the focus on tacit knowledge as opposed to explicit documentation, and the tension between informal and formal communication.
- Organization/management related challenges, such as the hierarchical, centralized decision making in plan-driven methods versus the empowerment of agile developers to make their own decisions.

In the organizations we studied, all three types of challenge did occur.

Collaboration and communication between the agile project members and externals is discussed in a number of studies [61,57,64,65,67]. Perreira et al. [61] report on a study of a mature agile team interacting with User Experience (UX) designers working on the same project. As in the organizations we studied, this requires cooperation between groups. Both groups can do their work independently, but at some point in time results need to be combined. This introduces dependencies between the groups, and problems may arise if both groups hold different *expectations* about the dependencies. Such was evidently the case in our organizations when it comes to customer involvement. Ferreira et al. further observe that managing the cooperation between the two groups requires articulation work: *"the specifics of putting together tasks, task sequences, task clusters - even aligning larger units such as lines of work and sub-projects - in the service or workflow"* [72]. The alignment practices discussed in Section 4.4.3 serve this purpose (in another context [73], these are referred to as boundary spanning activities).

In the case study reported in Pikkarainen et al. [64], the authors observe that the biggest communication hurdles in the project occurred between the agile development team and the external staff. Here, "external staff" refers to the technical staff who are involved in other projects; in the organizations we studied these would be the developers of the back-end systems. They observe, like others [14,51] that managing the interfaces between those groups is challenging. An observation we also make. In [65], communication and collaboration with stakeholders is identified as one of the major barriers in agile deployment.

In both companies studied, the agile initiative started bottom-up, i.e. the development teams are the first in the organization to implement agile practices. However, the transition from a plan-driven to an agile process affects not only the development team members, but also other teams, departments, and management [21]. From our data, it is clear that the remains of a plan-driven way of working complicate the communication between business and IT. It takes effort to change from a "we-them" attitude to a collaborative culture [11,21]. As a Scrum Master pointed out: *"Agile is not only about a change in process, but also a change in culture . . . and that requires time."* How an agile process is introduced to an organization will significantly impact the ultimate success of the process change [21]. We found that implementing an agile process within an existing project organization confuses both developers and business representatives. Unsurprisingly so, as business representatives are used to their involvement within plan-driven software development being typically limited to provisioning requirements at the beginning and providing feedback at the end of the process [79,15,38], and developers are used to the developer-centric approach of plan-driven development; there is limited to no regular interaction between development teams and their customers [29,34]. It is paramount to identify such model clashes [14], and take actions to mitigate their effect. The actions discussed in Sections 3.3 and 4.3 are examples of such mitigating actions. One would expect that a colleague-to-colleague relationship between enterprise departments should have its advantages over a supplier-to-customer relationship, however, our findings show no proof of that. In fact, the consequences of a lack of involvement discovered by us are highly similar to results of Hoda et al. [38], who investigated customer collaboration in the latter context.

Nerur and his colleagues also investigated the co-existence of agile and plan-driven IT development in one organization [58,81]. They discuss agile and plan-driven IT development in terms of two types of organizational innovation behaviors, exploitation and exploration. Quoting He and Wong [36], Nerur et al. [58] state that.

> "*In general, exploration is associated with organic structures, loosely coupled systems, path breaking, improvisation, autonomy and chaos, and emerging markets and technologies. Exploitation is associated with mechanistic structures, tightly coupled systems, path dependencies, routinization, controol and bureaucracy, and stable markets and technology.*"

To deal with both, they propose an ambidextrous organization, with two highly coupled subunits, the agile subunit and the plan-driven subunit, with a tight governance structure at the IT management level. This organizational structure is not yet recognizable in the companies we investigated. The exploitation and exploration characteristics are present though, and our research highlights the challenges these opposing characteristics bring to the organization.

[1] Customers of agile teams in our research are business representatives. Existing literature mainly refers to *customer involvement* or *customer collaboration*. In this article, these terms are interchangeable.

Some of the recurring problems observed in earlier studies are not present in the main categories we identified. Most notably, the lack of attention to architecture common in agile methods (see, e.g., [62,17,55]), and the challenges to realize a continuous testing regimes using agile approaches (see, e.g., [68,62,65]) do not show up in our analysis. The fact that architecture was not mentioned very often is probably due to the fact that we mostly interviewed people from the agile teams. Problems regarding testing were mentioned incidentally, but not often enough for the resulting codes to be taken into consideration in the categories for this study. At Company A, a reason for this could be the lack of a decent testing environment at the time the interviews were held. As one interviewee said:

> "*The environment is a pressing problem. I'm talking about workstations that are not fully up-and-running ... or a test environment that is simply missing. Continuous Integration, which is missing, is an invaluable asset for agile practice ... Although their working on it, Continuous Integration is still not finished. We need automatic regression tests for our sprints to be more efficient. We don't want to keep on testing manually, as our code grows our effort spent in testing will grow, as well as the risk of stuff falling apart at one end of the system when code is deployed at the other end.*" —Scrum Master

6. Threats to validity

A theory produced by Grounded Theory research cannot be claimed to be universally applicable. However, it can be modified by constant comparison and accommodation of more data from new contexts [31]. The remainder of this section presents other important limitations.

6.1. Participant selection

Regarding the participant selection procedure, we were highly dependent on managers at the participating companies. The participants selection occurred in a "one must make with what one has" fashion. We communicated our requirements to the participating companies, but not all practitioners were able – or willing – to participate. As a result, not all parts of the landscape are fully represented in the data. In particular, we mostly interviewed people from the agile projects, and not enough business representatives. As a consequence, the factors identified as well as the mitigating actions may be biased towards the agile point of view. In a similar vein, participants may have wanted to distance themselves from the traditional way of working, and act as "pure" agilists. This could make the challenges observed stronger than they are in many an organization. An advantage though is that it makes them more visible as well.

6.2. Data collection methods

Due to time and access constraints, data was only collected using semi-structured interviews. We were not allowed to participate in the daily process of the participants, therefore, we did not witness the events described first hand. As a result, this may provide a rather superficial description of the phenomena and accesses only what participants say they do. Moreover, data derived from interviews is known to be prone to bias [60]. Four types of data can be presented to the researcher: (1) Baseline data; the best description a participant can offer, (2) proper-line data; what the participant thinks is proper to tell the researcher, (3) interpreted data; what is told by a trained professional who wants to make sure that others see the data his' professional way, and (4) Vaguing it out; the vague information provided by a participant that is not bothered to provide information to the researcher [30]. All of these were identified during the process, often in retrospect, and were cleared up by communicating with the participant afterwards. However, some may have gone unnoticed.

6.3. Theoretical sampling

Grounded Theory's practice *Theoretical Sampling* dictates the researcher to fully analyse and code his interview before conducting the next interview. This proved impossible in our setting, we were forced to conduct our interviews in a few weeks. As a result, there was only time to transcribe part of an interview before the next one was held. However, while the actual coding was not conducted until after all interviews were completed, this concept was taken into consideration during the interviews at Company B, keeping in mind earlier interviews and deepening emerged phenomena. Results from Company A were coded and taken into consideration, before interviews were commenced at Company B.

Our study took place at two organizations only. We have come across several other organizations with similar characteristics in terms of combining plan-based and agile approaches, but have not done a similar study there.

7. Conclusions

This article presents a grounded theory of agile practices in traditional enterprises, based on a Grounded Theory research involving 21 agile practitioners from two large enterprise organizations in the Netherlands. The theory explains how enterprise factors *Increased IT Landscape Complexity* and *Lack of Business Involvement* result from the introduction of agile practices to an otherwise traditional development organization, and how practitioners cope with the challenges at hand.

We grouped our findings as follows:

- **Increased IT Landscape Complexity.** A large and complex IT landscape is inherently connected to enterprise software development. Years of evolution of this landscape used to support the business results in a collection of highly interconnected information systems. In our study, coping with the resulting complexity emerged as one of the most daunting tasks for agile practitioners in this context. Successful mitigation strategies used by the companies we investigated are: (1) creating a common sense of purpose, (2) manage alignment at the programme level, and (3) facilitate the role of the project manager.
- **Lack of Business Involvement.** During our research lack of business involvement was indicated as *"one of the hardest things to cope with when practising agile"*. Although many aspects of business involvement are similar to findings about customer involvement or customer collaboration in previous studies, some context specific phenomena were found that we did not come across in existing literature. Successful mitigation strategies identified are: (1) changing the mindset of business stakeholders, (2) channeling business knowledge through the Product Owner, and (3) aligning knowledge and requirements at the business level.

The two factors into which we organized our results are not new. They are in line with, and corroborate, findings of earlier studies, most of which concern the introduction of agile methods, or the transition from plan-based to agile methods, rather than the simultaneous use of both. Our research points out that the co-existence of agile and plan-driven methods brings the same challenges as agile adoption does. This need not be a surprise, since the adoption of agile practices induces the co-existence of both, at least for

a while. Some of the recurring problems mentioned in earlier studies were not mentioned in the interviews we have had. Most notably, the lack of attention to architecture that is common in agile methods, and the different testing regimes used by plan-driven and agile approaches, were not often mentioned by our respondents.

The contribution this paper brings is the identification of successful mitigation strategies to deal with the challenges of co-existing plan-driven and agile methods. The mitigating actions used in the organizations we studied all concern the communication between the agile and the plan driven part of the organizations, and the timing thereof. These actions can be understood as boundary spanning activities between the two communities.

Several themes worthy of further investigation caught our interest during this research.

- Traditional IT management vs. Agile development

We found that difference in culture slowed down the agile adoption progression. In both organizations, agile initiatives were adopted bottom-up, meaning that the agile practices were implemented at a development team level, with the intention to grow into the organization, "like a slowly expanding oil slick." An important observation was how IT management and development differed from each other regarding culture and work method. This was indicated as one of the causes for both *Increased IT Landscape Complexity* and *Lack of Business Involvement*. This co-relation between cause and consequences is very interesting to pursue further.

- Emphasis on perceived benefits

During this research we focussed on factors influencing agile development teams in an enterprise environment. The way practitioners coped with these factors of the environment emerged naturally. Our conclusion is that agile practices can coexist with plan-driven methods within the same organization. However, we weren't able to get a grounded view of the benefits and in what manner they contributed to the enterprises, as for instance discussed in Misra et al. [56]. The relatively early state of the adoption process at both companies (respectively 1 and 1.5 years) could have caused this.

- Boundary spanning role of mitigating actions

The mitigating actions identified serve to aid in the communication between the agile and the plan driven part of the organizations, and the timing thereof. They are a bridge (i.e. boundary spanning) between these communities. A more systematic and in-depth study of the role of boundary spanning activities and artefacts could help to better align the simultaneous use of plan-driven and agile methods.

Acknowledgements

We thank the participants from company A and B for their willingness to participate in this research. We thank Torgeir Dingsøyr and Philippe Kruchten for their insightful comments on an earlier version of this article.

References

[1] P. Abrahamsson, J. Warsta, M.T. Siponen, J. Ronkainen, New directions on agile methods: a comparative analysis, in: Proceedings of the 25th International Conference on Software Engineering. ICSE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 244–254.
[2] S. Adolph, W. Hall, P. Kruchten, Using grounded theory to study the experience of software development, Empirical Software Engineering 16 (4) (2011) 487–513.
[3] S. Adolph, P. Kruchten, W. Hall, Reconciling perspectives: a grounded theory of how people manage the process of software development, Journal of Systems and Software 85 (6) (2012) 1269–1286.
[4] R. Agarwal, V. Sambamurthy, Principles and models for organizaing the IT function, MIS Quarterly Executive 1 (1) (2002) 1–16.
[5] G. Allan, A critique of using grounded theory as a research method, Electronic Journal of Business Research Methods 2 (1) (2003).
[6] S. Augustina, B. Payna, F. Sencindiver, S. Woodcock, Agile project management: steering from the edges, Communications of the ACM 48 (12) (2005) 85–89.
[7] R. Baskerville, J. Pries-Heje, S. Madsen, Post-agility: what follows a decade of agility?, Information and Software Technology 53 (5) (2011) 543–555
[8] K. Beck, M. Beedle, A.V. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, Manifesto for Agile Software Development. Tech. Rep., 2001. <http://www.agilemanifesto.org/>.
[9] G. Benefield, Rolling out agile in a large enterprise, in: Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS 2008), IEEE Computer Society, 2008, p. 462.
[10] D.F. Birks, W. Fernandez, N. Levina, S. Nasirin, Grounded theory method in information systems research: its nature, diversity and opportunities, European Journal of Information Systems 22 (1) (2013) 1–8.
[11] B. Boehm, Get ready for agile methods, with care, Computer 35 (2002) 64–69.
[12] B. Boehm, A view of 20th and 21st century software engineering, in: Proceedings of the 28th International Conference on Software Engineering. ICSE '06, ACM, New York, NY, USA, 2006, pp. 12–29.
[13] B. Boehm, R. Turner, Balancing Agility and Discipline, Addison Wesley, 2002.
[14] B. Boehm, R. Turner, Management challenges to implementing agile processes in traditional development organizations, IEEE Software (2005) 30–39.
[15] B.W. Boehm, A spiral model of software development and enhancement, Computer 21 (1988) 61–72.
[16] A. Brown, S. Ambler, W. Royce, Agility at scale: economic governance, measured improvement, and disciplined delivery, in: Proceedings 35th International Conference on Software Engineering (ICSE 2013), IEEE Computer Society, 2013, pp. 873–881.
[17] L. Cao, K. Mohan, P. Xu, B. Ramesh, A framework for adopting agile development methodologies, European Journal of Information Systems 18 (4) (2009) 332–343.
[18] F.K. Chan, J.Y. Thong, Acceptance of agile methodologies: a critical review and conceptual framework, Decision Support Systems 46 (4) (2009) 803–814.
[19] A. Cockburn, Agile Software Development, Addison-Wesley Professional, 2001.
[20] A. Cockburn, J. Highsmith, Agile software development, the people factor, Computer 34 (11) (2001) 131–133.
[21] M. Cohn, D. Ford, Introducing an agile process to an organization, Computer (2003) 74–78.
[22] G. Coleman, R. O'Connor, Using grounded theory to understand software process improvement: a study of Irish software product companies, Information and Software Technology 49 (2007) 654–667.
[23] K. Conboy, Agility from first principles: reconstructing the concept of agility in information systems development, Information Systems Research 20 (3) (2009) 329–354.
[24] K. Conboy, B. Fitzgerald, The views of experts on the current state of agile method tailoring, in: T. McMaster, D. Wastell, E. Ferneley, J. DeGross (Eds.), Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda. IFIP International Federation for Information Processing, vol. 235, Springer, Boston, 2007, pp. 217–234.
[25] D. Denison, G. Spreitzer, Organizational culture and organizational development: a competing values approach, in: R. Woodman, W. Pasmore (Eds.), Research in Organizational Change and Development, vol. 5, JAI Press, 1991, pp. 1–21.
[26] T. Dingsøyr, S. Nerur, V. Balijepally, N. Brede Moe, A decade of agile methodologies: towards explaining agile software development, Journal of Systems and Software 85 (6) (2012) 1213–1221.
[27] J. Drobka, D. Noftz, R. Naghu, Piloting XP on four mission-critical projects, IEEE Software 21 (6) (2004) 70–75.
[28] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Information and Software Technology 50 (9-10) (2008) 833–859.
[29] S. Fraser, A. Martin, R. Biddle, D. Hussman, G. Miller, M. Poppendieck, L. Rising, M. Striebeck, The role of the customer in software development: the XP customer – fad or fashion?, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications OOPSLA '04, ACM, New York, NY, USA, 2004, pp. 148–150.
[30] B. Glaser, Theoretical Sensitivity: Advances in the Methodology of Grounded Theory, Sociology Press, Mill Valley, CA, 1978.
[31] B. Glaser, Basics of Grounded Theory Analysis: Emergence vs Forcing, Sociology Press, Mill Valley, CA, 1992.
[32] B. Glaser, Doing Grounded Theory: Issues and Discussions, Sociology Press, Mill Valley, CA, 1998.
[33] B.G. Glaser, A.L. Strauss, The discovery of grounded theory, The British Journal of Sociology 20 (2) (1967) 227.
[34] P.S. Grisham, D.E. Perry, Customer relationships and extreme programming, SIGSOFT Software Engineering Notes 30 (4) (2005) 1–6.

[35] J. Harvey, The abilene paradox: the management of agreement, Organizational Dynamics 3 (1) (1974) 63–80.
[36] Z. He, P. Wong, Exploration vs. exploitation: an empirical test of the ambidexterity hypothesis, Organization Science 15 (4) (2004) 481–494.
[37] J. Heidenberg, M. Matinlassi, M. Pikkarainen, P. Hirkman, J. Partanen, Systematic piloting of agile methods in the large: two cases in embedded systems development, in: M.A. Babar, M. Vierimaa, M. Oivo (Eds.), PROFES2010, LNCS, vol. 6156, Springer Verlag, 2010, pp. 47–61.
[38] R. Hoda, J. Noble, S. Marshall, The impact of inadequate customer collaboration on self-organizing agile teams, Information and Software Technology 53 (5) (2011) 521–534.
[39] R. Hoda, J. Noble, S. Marshall, Developing a grounded theory to explain the practices of self-organizing agile teams, Empirical Software Engineering 17 (6) (2012) 609–639.
[40] R. Hoda, J. Noble, S. Marshall, Self-organizing roles on agile software development teams, IEEE Transactions on Software Engineering 39 (3) (2013) 422–444.
[41] J. Iivari, N. Iivari, The relationship between organizational culture and the deployment of agile methods, Information and Software Technology 53 (5) (2011) 509–520.
[42] M. Isham, Agile Architecture IS Possible You First Have to Believe! Agile 2008 Conference, 2008, pp. 484-489.
[43] I. Janis, Victims of Groupthink, Houghton Mifflin Company, 1972.
[44] D. Karlström, P. Runeson, Integrating agile software development into stage-gate managed product development, Empirical Software Engineering 11 (2) (2006) 203–225.
[45] P. Kettunen, M. Laanti, Combining agile software projects and large-scale organizational agility, Software Process Improvement and Practice 13 (2) (2008) 183–193.
[46] P. Kruchten, Contextualizing agile software development, Journal of Software: Evolution and Process 25 (4) (2013) 351–361.
[47] N. Kurapati, V.S.C. Manyam, K. Petersen, Agile software development practice adoption survey, in: Proceedings XP 2012, LNBIP, vol. 111, Springer Verlag, 2012, pp. 16–30.
[48] M. Laanti, O. Salo, P. Abrahamsson, Agile methods rapidly replacing traditional methods at Nokia: a survey of opinions on agile transformation, Information and Software Technology 53 (2011) 276–290.
[49] C. Larman, B. Vodde, Lean Primer, 2009. <http://www.leanprimer.com/downloads/lean_primer.pdf>.
[50] N. Levina, E. Vaast, The emergence of boundary spanning competence in practice: implications for implementation and use of information systems, MIS Quarterly 29 (2) (2005) 335–363.
[51] M. Lindvall, D. Muthig, A. Dagnino, C. Wallen, M. Stupperich, D. Kiefer, J. May, T. Kähkönen, Agile software development in large organizations, IEEE Computing Practices (2004) 26–34.
[52] A. Magdaleno, C. Werner, R. de Araujo, Reconciling software development models: a quasi-systematic review, The Journal of Systems and Software 85 (2) (2012) 351–369.
[53] R. Matavire, I. Brown, Profiling grounded theory approaches in information systems research, European Journal of Information Systems 22 (1) (2013) 119–129.
[54] J. McAvoy, T. Butler, The role of project management in ineffective decision making within agile software development projects, European Journal of Information Systems 18 (4) (2009) 372–383.
[55] D. Mishra, A. Mishra, Complex software project development: agile methods adoption, Journal of Software Maintenance and Evolution: Research and Practice 23 (8) (2011) 549–564.
[56] S.C. Misra, V. Kumar, U. Kumar, Identifying some important success factors in adopting agile software development practices, The Journal of Systems and Software 82 (2009) 1869–1890.
[57] E. Moore, J. Spens, Scaling agile: finding your agile tribe, in: Agile 2008 Conference, IEEE Computer Society, 2008, pp. 121–124.
[58] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Communications of the ACM 48 (5) (2005) 73–78.
[59] OGC, An Introduction to PRINCE2: Managing and Directing Successful Projects. The Stationary Office, 2009.
[60] K.W. Parry, Grounded theory and social process: a new direction for leadership research, The Leadership Quarterly 9 (1) (1998) 85–105.
[61] J. Perreira, H. Sharp, H. Robinson, User experience and agile development: managing cooperation through articulation work, Software – Practice and Experience 41 (9) (2011) 963–974.
[62] K. Petersen, C. Wohlin, A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case, Journal of Systems and Software 82 (9) (2009) 1479–1490.
[63] K. Petersen, C. Wohlin, The effect of moving from a plan-driven to an incremental software development approach with agile practices, Empirical Software Engineering 15 (6) (2010) 654–693.
[64] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still, The impact of agile practices on communication in software development, Empirical Software Engineering 13 (3) (2008) 303–337.
[65] M. Pikkarainen, O. Salo, R. Kuusela, P. Abrahamsson, Strengths and barriers behind the successful agile deployment – insights from the three software intensive companies in Finland, Empirical Software Engineering 17 (6) (2012) 675–702.
[66] D.H. Pink, Drive: The Surprising Truth About What Motivates Us, Riverhead Books, 2011.
[67] J. Rasmussen, Introducing XP into Greenfield projects: lessons learned, IEEE Software 20 (3) (2003) 21–28.
[68] H. Robinson, H. Sharp, Organisational culture and XP: three case studies, in: Proceedings of the Agile Development Conference (ADC'05), IEEE, 2005.
[69] K. Schwaber, J. Sutherland, The Scrum Guide: The Official Rulebook. Tech. Rep, October 26, 1991. <http://www.scrum/scrumguides/, http://www.scrum.org/scrumguides/>.
[70] K.V. Siakas, E. Siakas, The agile professional culture: a source of agile quality, Software Process Improvement and Practice 12 (6) (2007) 597–610.
[71] S.L. Star, J.R. Griesemer, Institutional ecology, 'Translations' and boundary objects: amateurs and professionals in berkeley's museum of vertebrate zoology, 1907–1939, Social Studies of Science 19 (3) (1989) 387–420.
[72] A. Strauss, The articulation of project work: an organizational process, The Sociological Quarterly 29 (2) (1988) 163–178.
[73] D.E. Strode, S.L. Huff, B. Hope, S. Link, coordination in co-located agile software development projects, Journal of Systems and Software 85 (6) (2012) 1222–1238.
[74] D.E. Strode, S.L. Huff, A. Tretiakov, The impact of organizational culture on agile method use, in: Proceedings of the 42nd Hawaii International Conference on System Sciences, IEEE Computer Society, 2009, pp. 1–9.
[75] J. Sutherland, Agile can scale: inventing and reinventing SCRUM in five companies, Cutter IT Journal 14 (2001) 5–11.
[76] H. Svensson, M. Höst, Introducing an agile process in a software maintenance and evolution organization, in: Proceedings of the 9th European Conference on Software Maintenance and Reengineering (CSMR'05), IEEE Computer Society, 2005.
[77] M.L.M.C. Tanis, The enterprise system experience – From adoption to success. In: R.W. Zmud, M.F. Price, (Eds.), Framing the Domains of IT-Management: Projecting the Future Through the Past, Pinnaflex Educational Resources, 2000, pp. 173–207 (Chapter 10).
[78] C. Tolfo, R.S. Wazlawick, The influence of organizational culture on the adoption of extreme programming, Journal of Systems and Software 81 (11) (2008) 1955–1967.
[79] H. vanVliet, Software Engineering: Principles and Practice, third ed., John Wiley, 2008.
[80] L. Vijayasarathy, D. Turk, Drivers of agile software development use: dialectic interplay between benefits and hindrances, Information and Software Technology 54 (2) (2012) 137–148.
[81] V. Vinekar, C.W. Slinkman, S. Nerur, Can agile and traditional systems development approaches coexist? An ambidextrous view, Information Systems Management 23 (3) (2006) 31–42.