UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Zoran Nebić

Agilni razvoj programske opreme v plansko vodenih organizacijah

MAGISTRSKO DELO

Ljubljana, 2016

UNIVERSITY OF LJUBLJANA
FACULTY OF COMPUTER AND INFORMATION SCIENCE

Zoran Nebić

Agile software development in a context of plan-based organizations

MASTER THESIS

Advisor: Professor Viljan Mahnič, PhD

Ljubljana, 2016

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Zoran Nebić

Agilni razvoj programske opreme v plansko vodenih organizacijah

MAGISTRSKO DELO

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2016

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Večna pot 113
1000 Ljubljana, Slovenija
telefon: 01 47 98 100
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.

Številka: 156-MAG-ISO/2016
Datum: 06. 04. 2016

Zoran NEBIĆ, prof. inf., ang. in knjiž.

L j u b l j a n a

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Agilni razvoj programske opreme v plansko vodenih organizacijah**

Agile software development in a context of plan-based organizations

Tematika naloge:

Agilni razvoj programske opreme se v zadnjih petnajstih letih vedno bolj uporablja s ciljem izboljšanja tradicionalno zamudnega in uporabniku dokaj neprijaznega procesa razvoja programske kode. Ker so posledice agilnega razvoja in njegovega vpliva na zaposlene še vedno nejasne, je pomembno razumeti koristi, priložnosti in omejitve tega razvoja oziroma mehanizma sodelovanja. Posledično je zbiranje vplivnih empiričnih dokazov za odločevalce na področju poslovne politike in razvoja programske opreme odprto raziskovalno področje.

Magistrsko delo vsebuje analizo možnosti agilnega razvoja programske opreme in kako se ta pristop lahko uporabi pri razvojnih procesih v podjetjih z namenom večje učinkovitosti, krajšega časa dostopa do trga, kot tudi večje ustreznosti razvitih proizvodov ali storitev strankam. Z raziskovanjem ključnih značilnosti različnih metod in postopkov, so analizirane možnosti in omejitve izbranih pristopov ter povezane z nedavnimi literarnimi vpogledi.

M e n t o r :

prof. dr. Viljan Mahnič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

D e k a n :

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

magistrskega dela

Spodaj podpisani/-a Zoran Nebić,

z vpisno številko 63080481,

sem avtor/-ica magistrskega dela z naslovom

Agilni razvoj programske opreme v plansko vodenih organizacijah

Agile software development in a context of plan-based organizations

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal/-a samostojno pod vodstvom mentorja (naziv, ime in priimek)

  prof. dr. Viljan Mahnič

- so elektronska oblika magistrskega dela, naslova (slov., angl.), povzetka (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko magistrskega dela

- in soglašam z javno objavo elektronske oblike magistrskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 31.08.2016. Podpis avtorja/-ice: 

i. Acknowledgements

I would like to thank my advisor, Professor Viljan Mahnič for guiding and supporting me over the years. You have set an example of excellence as a mentor and instructor, getting me back on track after I almost gave up.

I would like to thank my fellow students, collaborators and colleagues who contributed to this research. Without you this thesis would have never been possible.

I would like to thank my family for the love, support and constant encouragement I have gotten over the years. In particular I would like to thank my dad. Without you I would have never restarted this research, I know it meant a lot to you. I would also like to thank my sister, my brother and Biserka: each one of you help me, in your own way, in everything I do.

I would also like to thank all my amazing friends, both old and new. You have helped shape me into what I am today, and will continue doing that in the future. In particular I would like to thank Milica. You have been a great encouragement and inspiration in the past couple of years and I hope will be in many years to come. I would also like to thank Vitomir. You are not only my friend, but also my chosen family.

ii. Dedication

I would like to dedicate this thesis to my dear mother. It was you who helped build the intellectual curiosity that brought me to the world of academia in the first place. Even though my life is on a different path now, critical thinking and open-mindedness remained. Thank you!

iii. Index

i. Acknowledgements v
ii. Dedication v
iii. Index vi
iv. List of Figures viii
v. List of Tables viii
vi. Abstract ix
vii. Povzetek magistrske naloge ix
a. Prispevek k trenutni raziskavi ix
b. Ključni rezultati iz teorije x
c. Ključni rezultati iz prakse xi
d. Omejitve raziskave xii
e. Nadaljnje raziskave in obeti xiii
1 Introduction 1
1.1 Motivation 1
1.2 Expected Results 2
1.3 Structure of the Thesis 2
2 Theoretical Background 4
2.1 Software Development in Practice 4
2.2 Heavyweight Models 5
2.2.1 Waterfall 5
2.2.2 Discussion of Waterfall 9
2.3 Agile Development Methods 10
2.3.1 Scrum 10
2.3.2 Kanban 13
2.3.3 Lean Software Development 14
2.3.4 Extreme Programming (XP) 15
2.3.5 Feature-Driven Development (FDD) 16
2.4 Conclusions 16
3 Waterfall versus Agile Development Methodology 18
3.1 Chapter Introduction 18
3.2 Company A: Agile / Scrum 18
3.3 Company B: Agile turned waterfall 20
3.4 Summary of Practice 23

4 Case Study and Results 24
4.1 Research Questions 24
4.2 Research Tools 24
4.3 Data Collection and Analysis 25
4.4 Main Results 25
4.4.1 Demographics 26
4.4.2 General employee satisfaction 29
4.4.3 Waterfall development experiences 32
4.4.4 Perceived satisfaction during waterfall projects 34
4.4.5 Agile development experiences 35
4.4.6 Perceived satisfaction during agile projects 38
4.5 Conclusions 40
5 Summary of the Thesis 43
6 References 45
7 Appendix 50
7.1 Questionnaire 50

iv. List of Figures

Figure 1: Implementation steps to develop a large computer program for delivery to a customer. (Adopted from [42]) 6
Figure 2: Critical steps and iterations during Waterfall development [42] 6
Figure 3: The ideal iterative interaction between phases during Waterfall [42] 7
Figure 4: Sprint Cycle (Adopted from [44]) 11
Figure 5: Lean approach to building integrity (Adopted from [45]) 15
Figure 5: Age distribution of the respondents 26
Figure 6: Age distribution comparison between Company A and Company B 27
Figure 7: Highest level of education 27
Figure 8: Duration of company affiliation 28
Figure 9: Familiarity of technologies 28
Figure 10: Responses to the question "I feel valued as an employee" 29
Figure 11: Clarity of job requirements comparison 30
Figure 12: Adequacy of job training 30
Figure 13: Job satisfaction comparison between Companies A and B 31
Figure 14: Job satisfaction across all the respondents 32
Figure 15: Distribution of roles in waterfall methodology within the companies 33
Figure 16: Maturity of agile methodology components (higher is better) 34
Figure 17: Satisfaction while working in waterfall methodologies 35
Figure 18: Roles performed within agile methodology practice 36
Figure 19: Relative maturity level 37
Figure 20: Perceived satisfaction in Agile projects – combined data for both companies 38
Figure 21: Perceived satisfaction in Agile projects 39

v. List of Tables

Table 1: Rough comparison between Waterfall and Agile Development 10

vi. Abstract

Agile software development has increasingly been used in the last fifteen years with the goal of improving traditionally time-consuming and rather non-user friendly process of developing software code. As implications of agile development and its impact on employees are still unclear, it is important to understand the benefits, opportunities and limitations of this development or collaboration mechanism. Thus, empirical evidence with implications for decision makers in the field of corporate policy and software development is an open research field.

This master thesis analyzes the potentials of agile software development and how this approach can be used to support the development processes in companies, in terms of efficiency, shorter time-to-market as well as better customer fit of the developed products or services. By exploring some of the key features of different methods and processes, the potentials and limitations of the selected approaches are analyzed and linked to recent literature insights.

Keywords: *agile development, Scrum, Kanban, corporate organizations, financial services*

vii. Povzetek magistrske naloge

To poglavje vključuje povzetek magistrske naloge ter teoretične in praktične učinke, omejitve in predloge za nadaljnje raziskave.

a. Prispevek k trenutni raziskavi

Odkar so se pred petnajstimi leti začele uvajati agilne metodologije, se razvoj programske opreme premika od tradicionalnega, zaporednega poteka dela k bolj fleksibilnim možnostim ter dvosmernim ali vzporednim procesom – še zlasti v podjetjih.

Poleg izboljševanja dragih, zamudnih in uporabniku neprijaznih procesov dostave programske opreme je namen agilne metode razvoja programske opreme povečati fleksibilnost in prilagodljivost ter se bolje prilagajati tržnim razmeram in povpraševanju strank.

Medtem ko je pred petnajstimi leti razvoj programske opreme potekal tako, da je ta lahko ostala nespremenjena deset let, se danes izdaje programske opreme ali zahteve spreminjajo zelo hitro. Ker so posledice agilnega razvoja in njegov vpliv na podjetja in razvojne ekipe še zmeraj neznane, smo opravili empirično raziskavo, da bi razumeli prednosti, priložnosti, tveganja ter omejitve agilnega razvoja in mehanizmov sodelovanja.

Ker nismo našli dovolj empiričnih dokazov z jasnimi usmeritvami za odločevalce s področja politike gospodarskih združb in razvoja programske opreme, smo s svojim delom prispevali k upravljanju agilnih metodologij v primerjavi s tradicionalnimi procesi, da bi raziskali in razumeli odprto področje raziskav.

Ta magistrska naloga vključuje analizo zmožnosti agilnega razvoja programske opreme in kako lahko ta pristop uporabimo za podporo procesom razvoja v podjetjih s ciljem večanja učinkovitosti, skrajševanja časa do prodaje na trgu in boljše prilagojenosti razvitih izdelkov in storitev strankam. Z raziskovanjem ključnih lastnosti različnih metod in procesov smo se osredinili na vzpostavitev procesov agilnega razvoja v praksi podjetja. To smo povezali s teorijo in najnovejšimi dognanji iz literature.

b. Ključni rezultati iz teorije

Razvoj programske opreme je ena najmočnejših industrij, ki po eni strani ustvari vse več služb in podjetij ter po drugi veliko raziskovalnih tem. Razvoj programske opreme je danes ena od glavnih dejavnosti v podjetjih, pa naj gre za vzdrževanje obstoječih programskih rešitev in sistemov za potrebe vsakodnevnega operativnega poslovanja ali za posodabljanje poslovanja z uvajanjem novih orodij in digitalizacijo podjetja z novimi tehnologijami. Metodologije razvoja programske opreme so v obeh primerih izjemnega pomena.

Za empirično raziskavo smo izbrali bančno industrijo, torej uveljavljeno industrijo, v kateri trenutno potekajo racionalne spremembe. Zato lahko sklepamo, da je tudi razvoj programske opreme temeljni del procesa sprememb. Da bi pridobili celovit vpogled v to neraziskano področje, raziskava temelji na kvalitativnih metodah raziskovanja, kar vključuje empirični vprašalnik in ankete s strokovnjaki v primerjalni študiji primera.

Prvi korak magistrske naloge je bil pridobiti znanje o razvoju programske opreme v teoriji, kot je opisano v literaturi. Vodilni programski inženirji teorijo razvoja programske opreme pogosto opisujejo s povsem tehničnega vidika. Akademski svet vsekakor zavzema analitični pristop pri razlaganju novih metod pri razvoju programske opreme.

Na primer tradicionalni proces razvoja programske opreme oz. tako imenovani slapovni model je mogoče primerjati z vidiki vodenja projekta (programske opreme). Po drugi strani pa so izbrane agilne metodologije, ki jih obravnavamo in ki vključujejo Scrum, Kanban, vitek razvoj programske opreme, ekstremno programiranje ter funkcijsko voden razvoj programske opreme, obrazložene s procesnimi koraki, tehnologijami, dostavo rezultatov in metodami sodelovanja.

Prvi del te magistrske naloge vključuje pregled najnovejše literature in empirične študije primerov iz zadnjih dveh desetletij. To poglavje vključuje različna dognanja iz začetnega raziskovanja te teme in iz pregleda literature. Zajema tudi definicije plansko vodenega razvoja programske opreme in spremembe proti modernim in agilnim metodologijam razvoja.

Natančneje, 2. poglavje vključuje razprave o Scrumu (2.3.1), Kanbanu (2.3.2), vitkem razvoju programske opreme (2.3.3), ekstremnem programiranju (2.3.4), funkcijsko vodenem razvoju programske opreme (2.3.5) in slapovni metodi (2.2.1).

Med začetnim raziskovanjem literature smo pridobili naslednje ključne ugotovitve:

1. Slapovni ali tradicionalno strukturirani procesi razvoja so močnejši pri opredelitvi zahtev, infrastrukture, vmesnikov, in kar je najbolj pomembno – pričakovanj in funkcij za stranke na začetku razvoja.
2. V poznejših fazah so stroški in napor zaradi spreminjanja zahtev v slapovni metodi zelo visoki. Testiranje in povratne informacije od stranke pa dobimo šele v poznejši fazi procesa.
3. V nasprotju s slapovno metodo začetna vzpostavitev agilnega procesa zahteva precej truda in časa, da proces dostave nove programske opreme postane učinkovit proces z različnimi vlogami in odgovornostmi.
4. Vendar ko je agilni proces vzpostavljen, je dostava funkcij in povratnih informacij hitrejša, stranka je lahko vključena v odločanje, hkrati pa se zmanjša kompleksnost razvoja zaradi sprotne dostave posameznih inkrementov končnega izdelka.

c. Ključni rezultati iz prakse

Kot je mogoče videti v raziskavi trenutne literature o digitalnem bančništvu, je bilo pri bančnih inovacijah ali agilnem razvoju (2. poglavje) opravljenih veliko raziskav, ki se navezujejo na proizvod, projekt ali težave med procesom. Hkrati pa trenutno stanje raziskav ponuja veliko odprtih vprašanj, na primer: katera metodologija razvoja programske opreme se prednostno uporablja v bančništvu ali kako izbira metodologije vpliva na zadovoljstvo članov ekipe, kakovost razvoja in zrelost procesa.

Da bi odgovorili na zastavljena vprašanja, smo si v 3. poglavju natančneje ogledali dva praktična primera razvoja programske opreme v bančništvu. Podjetje A je primer agilnega razvoja, medtem ko se podjetje B ukvarja s prehodom iz agilne metode v slapovno metodo. Analizirali smo agilno metodologijo, ki je bila zastavljena v razmeroma neprijaznem okolju, poleg tega smo si ogledali spremembo metode iz agilne v slapovno v občutno manj strukturiranem in definiranem okolju. Na koncu smo oba primera primerjali in postavili konkretna raziskovalna vprašanja za primerjavo teorije s praktičnimi primeri.

Čeprav se je v podjetju A zdelo, da z agilno metodologijo vse poteka, kot je treba, in je tesno sledila teoriji in praksi metode Scrum, je bil projekt ukinjen po prvi fazi zbiranja sredstev. Ko so pri razvojni in vodstveni ekipi nastale težave pri promociji projekta v velikem podjetju in pri iskanju nadaljnjega financiranja, se je proces razvoja upočasnil. Rezultat te inovativne iniciative je bila povsem razvita aplikacija – delujoča in integrirana v sistem podjetja –, vendar je bila kljub temu po končanem razvoju redko uporabljena v praksi.

Čeprav so se v podjetju B metodologije, strukture, finančni in drugi sporazumi ves čas spreminjali, je okolje ostalo osredotočeno na pomembnost razvoja proizvoda, na njegovo kakovost in funkcionalnost. Grožnje temu projektu so bile bolj notranje kot zunanje težave.

Na podlagi analize obeh primerov smo opravili anketo za člane ekipe v podjetju A in podjetju B. Podrobnejša analiza rezultatov ankete in grafični prikaz podatkov sta predstavljena v 4. poglavju.

Ključni rezultati:

a) Agilni razvoj je mogoče uspešno uporabiti v bančnem sektorju.

b) V določenih primerih se je morda primerno vrniti k slapovni metodi.

c) Zadovoljstvo članov ekipe v podjetju se odraža v projektih.

d) Medtem ko pravilna uporaba metodologije (agilne ali druge) ni garancija za uspešnost projekta (podjetje A), njena odsotnost ali nestabilnost (podjetje B) lahko povzroči dodatne težave v že tako občutljivem in zapletenem procesu razvoja programske opreme.

Študija primerov je še pokazala, da teoretičnih konceptov ni zmeraj enostavno prenesti v prakso. Pri tem bi lahko bila glavna težava ta, da se velika podjetja ves čas spoprijemajo s spremembami in ne zmorejo zasnovati procesa dostave programske opreme iz nič, temveč morajo prilagoditi trenutni model dostave (ali pogosto več modelov, ki potekajo hkrati).

Izbira kvalitativne metode za naše raziskovalno delo in še zlasti to magistrsko nalogo je bila zelo dragocena, saj je podala zelo zanimiv vpogled v praktične primere. Empirične raziskave v industriji in teorija, ki vključuje tradicionalni razvoj programske opreme, gredo z roko v roki na področju metodologij agilnega razvoja.

Zato predlagamo nadaljevanje raziskav, saj pričakujemo, da se bodo nenehno dogajala nova odkritja in rezultati za razvojne ekipe, strategе, odločevalce, tj. podjetja na splošno.

Na splošno velja, da bodo omejitve pri implementaciji novih konceptov in sprememb pri glavnih procesih vedno vodile do kompromisov. Če se osredinimo samo na pravilno integracijo standardiziranega procesa namesto na temeljne vrednote, lahko ogrozimo tudi vzpostavitev agilne metodologije. Pri tem smo ugotovili, da če vsi pomembni deležniki, torej člani ekipe, razumejo uporabljeno metodologijo, lahko hitro zavzamejo vloge in odgovornost, ter najpomembnejše – lahko delajo kot ekipa.

d. Omejitve raziskave

Čeprav to raziskovalno delo ponuja pester nabor odgovorov na raziskovalna vprašanja, se zavedamo tudi omejitev pri svojem delu:

- Izbrali smo kvalitativno metodo raziskave, kar pomeni, da na svoja vprašanja odgovarjamo z opisnimi sklepi in ne s kvantitativnimi podatki.

- Čeprav je bila anonimnost udeležencev v anketi zagotovljena, so bili zelo zadržani pri kritiziranju podjetja. To je lahko razlog za pozitivne odgovore o zadovoljstvu ekipe.
- Raziskovalno delo se je osredotočalo na proces razvoja in ne na njegove rezultate.
- Zaradi nadaljnjih omejitev, kar se tiče časa in podatkov, se zavedamo, da naši rezultati niso univerzalni.

Ker so nekateri analizirani podatki notranji podatki podjetij, ta magistrska naloga ne podaja popolnih odgovorov na raziskovalna vprašanja, temveč ponuja vpogled v primere podjetja v srednji in vzhodni Evropi. Na odprto vprašanje, denimo, katera metoda je boljša za upravljanje deležnikov ali izkušnjo stranke, ni bilo mogoče odgovoriti.

e. Nadaljnje raziskave in obeti

V nadaljnjih raziskavah bi bilo zanimivo primerjati ta dva primera v zvezi z učinkovitostjo procesa, številom programskih napak, velikostjo ekipe, stroški razvoja, investicijo, strategijo in perspektivo stranke.

Ne glede na to, ali so razvoj programske opreme in povezane metodologije v povezavi z bančništvom analizirani v akademskem, komercialnem ali družbenem kontekstu, si ta tema zasluži nadaljnjo raziskavo.

1 INTRODUCTION

1.1 Motivation

The Agile Manifesto emerged in 2001 promising a revolution in the area of software development, leading to faster time-to-market, better software and, ultimately, happier customer and developer. It has brought "unprecedented changes to the software engineering field", and led to introduction of many software methods, tools, techniques and best practices [23]. Today, fifteen years into this practice, the topic remains fresh and it is still debatable whether agile methodologies fit all the industries and types of projects.

One example of an established traditional industry which is currently going through a rational change – whether in terms of digitalization of previously analogue processes, or disruptively novel services to customers, such as video chatting, crowdfunding or bitcoins – is banking. In essence it is perceived as completely opposite to modern establishments in the idea of a startup companies.

While a startup is built from a greenfield, not being afraid to be disruptive, different, and following this Silicon Valley mantra: "Fail Fast, Fail Often" [8], banks are perceived as being pillars of stability, and also a topic of special interest in the aftermath of 2008 banking crisis.

Although most of such traditional or large organizations have similar needs to change and adapt to market influences, they all need compelling evidence before adopting new methods and technologies and deploying it on a larger scale because of their size and complexity [34]. In most cases, new technologies and processes are not exchanged, but integrated with existing ones. Hence, the costs invested into changes are higher, the complexity of different processes and systems rises and in particular, the aspect of people getting used to their new roles and responsibilities can cause further delays in development and delivery in product development.

The banking sector is also well known to rely on large and monolithic legacy systems [18], developed in outdated and hard to maintain programming languages [48], not suitable for agile development [16], and having to catch up with rapid advancements in software development.

The mentioned characteristics make banking a perfect industry for observing agile methodologies in practice, and this thesis will be based on personal experience from two different multinational banking groups, with varying levels of project complexity and agile acceptance level.

While case studies that analyze agile methodologies and provide an insight into software development are existent in India [16], Pakistan [52], the US [12], [33], or Scandinavian countries [28], [38], this thesis contributes to research by providing an overview of agile practice in the CEE region, which seems interesting but not yet sufficiently investigated.

1.2 Expected Results

Even prior to its "official" start in 2001 agile technologies have been very interesting to the practitioners, resulting in sharp rise of published papers from 2000 to 2003, then being picked up by rising numbers of research papers starting from 2003 through 2005 [24]. Considerable research has been conducted with relation to product, project or process issues [21], but still leaving many topics unanswered, specifically adaptability and extension of agile methods, and, somewhat surprisingly also fundamental topics such as "what constitutes agility" [1].

This thesis aims to contribute to the field by providing analysis of potentials of agile in traditional, plan-based companies, and its results in practice. In this context several approaches will be evaluated for introducing agile teams in more traditional structure [28] [12]. The study will also be investigating the emerging change from an individual work towards self-managing teams, and providing an explanation to why this shift requires a change in corporate processes, systems, collaborations tools, and above all, a reorientation not only by developers but also by the management [36].

In reference to current findings in literature, agile methodologies have an impact on both quality and quality improvements. Following the literature stream of experiments and case studies that compare agile and waterfall development methods, this thesis includes both theoretical investigations and frameworks as well as empirical evidence to provide implications for software development practitioners in large enterprises.

The theoretical background in the first part of the thesis includes a literature review on definitions of agile methodologies, including SCRUM, Kanban, i.e. in particular. The second part of the thesis includes the research framework, arguments and survey questionnaires that were derived from relevant case studies. The third part of the thesis is an in-depth analysis of the conducted survey to other literature insights. The aim is to develop basic implications and prospects of agile methodologies in banking industry based on the individual survey experience and results.

1.3 Structure of the Thesis

Since the introduction of agile software development in 2001, software development – especially in corporate settings – is increasingly shifting from traditional, rather sequential workflows towards more flexible options and bidirectional or parallel processes. Obviously the main reasons for such change are reaching from an increase in delivery flexibility, to a higher customer satisfaction and adaptable project costs.

The first step in this thesis is to understand the software development practice in theory, as it is described in literature. On the one hand, software development theory is mostly discussed by some of the leading corporate software engineers from a rather technical perspective. On

the other hand, academia seems to have a rather analytical attitude to explain the new methods in software development. *Chapter Theoretical Background* includes an overview of the latest literature and empirical case studies from the last two decades. Various insights that result from the initial research of this topic and a literature review are discussed in this chapter. This includes definitions of heavyweight software engineering, as well as the emerging changes towards modern, agile development methodologies. In particular, Chapter 2 includes a discussion of Scrum (2.3.1), Kanban (2.3.2), Lean Software Development (2.3.3), Extreme Programming (2.3.4), and Feature-driven Development (2.3.5) next to the Waterfall method (2.2.1).

The practical experience of both traditional and agile development methodologies is discussed in form of an empirical case study in *Chapter Waterfall versus Agile Development Methodology*. Hereby, two practical examples from corporate settings from the CEE region are examined. In order to understand exemplary set-ups or organizations of software development teams in the banking sector, this part of the thesis focuses on discussing Waterfall and agile development from a practical point of view.

At the beginning of the study the question arises, whether the discussed methodologies are implemented in alignment with theoretical assumptions, or to what extent they were adapted or modified. Particularly corporate or managerial expectations to this case study would include an answer to the question of which methodology is most efficient or brings the best results. Since agile development is increasingly moving into the mainstream, this chapter includes some basic implications for choosing the appropriate form of software development for corporate settings. The chapter concludes with comparative arguments of both methodologies.

In order to underline the case study with empirical data, *Chapter Case Study and Results* contains the research framework, including concrete research questions (4.1) that were designed to look into the described cases by providing a qualitative questionnaire or expert interviews to team members. Further, this chapter includes a description of the research process, including the used tools (4.2), as well as data collection and analysis steps (4.3). Eventually, Chapter 4 concludes with the main results of the study (4.4) and conclusions to the conducted research (4.5).

A summary of the thesis, as well as an outlook into further relevant problem statements or open research topics that could not be included into this research work due to the scope of the thesis are discussed in *Chapter Summary of the Thesis.* The remainder of the thesis includes a reference list and the questionnaire that was used for the case study.

2 THEORETICAL BACKGROUND

2.1 Software Development in Practice

According to current literature, the process of software development is strongly linked to the process of project management, or IT-project management in particular. In reference to [41], organizations should develop and follow a well-defined project management process in order to achieve the best delivery results. Furthermore, the main reasons why projects fail are related to poor definitions or planning of requirements, resources, schedules, or unpredictable risks. In order to mitigate such risks, a shift from a traditionally organized software development process towards a flexible and adaptable process, in terms of product development, is expected.

Since the traditionally organized software development teams are said to accurately understand their customers' needs and document these in requirement lists or feature descriptions at the beginning of a development process, it is expected that the efforts for feature changes and bug fixing will be minimal [45]. In other situations, i.e. when: a) it is not possible to define the project scope or subordinate features at the beginning of the project, or b) a customer feedback is needed for developing functional or visual details, an agile methodology [11] can provide better results in terms of time-to-delivery, customer satisfaction, or to some extent even team satisfaction. This thesis aims at exploring particularly such differences or similarities between traditional and agile development methodologies.

In order to understand, distinguish and compare different methodologies, this chapter focuses on basic definitions and limitations of relevant software development methods. Based on the literature discussion in this chapter, the projects or processes which are selected for the empirical study are later classified into waterfall or agile methods and compared along the research questionnaires.

2.2 Heavyweight Models

Traditional ways of software development are specified as heavyweight methodologies in literature. They are based on a rather sequential process of development, including phases of definition, design, coding, testing and eventually implementation of the software.

This thesis will focus on agile development and the benefits towards heavyweight methodologies. Yet, in order to understand and determine the two ways of software production, this section includes a discussion about traditional software engineering based on the example of a structured, heavyweight method, - the Waterfall model. It is followed by a discussion of common agile practices: Scrum (2.3.1), Kanban (2.3.2), Lean Software Development (2.3.3), Extreme Programming (2.3.4), and Feature-driven Development (2.3.5) next to the Waterfall method (2.2.1).

2.2.1 Waterfall

Waterfall development was initially described by Winston Royce [42] in 1970 based on his experiences with the development of large software systems for the aircraft industry. Regardless of size or complexity, Royce coins two essential steps of software development: analysis and coding. In case of very simple implementation concepts, Royce narrows down the process of small software development into exactly these two steps. Particularly if the final product is to be operated by the developers themselves, this system seems both efficient and intact.

However, organizing large software manufacturing only along the two phases is critical and even *"doomed to failure"* [42]. Even though some of the additional development steps are not going to contribute to the final product directly or at all, because of the larger size and complexity, the development process needs to be extended.

Therefore, the process to support large-scale or complex software development which is delivered to a customer can be described as following: 1) define systems requirements, 2) define software requirements, 3) profound analysis, 4) program design, 5) coding, 6) testing and eventually 7) operations. The following figure illustrates the sequential process. By definition, each phase is succeeded by the next unidirectional process step.

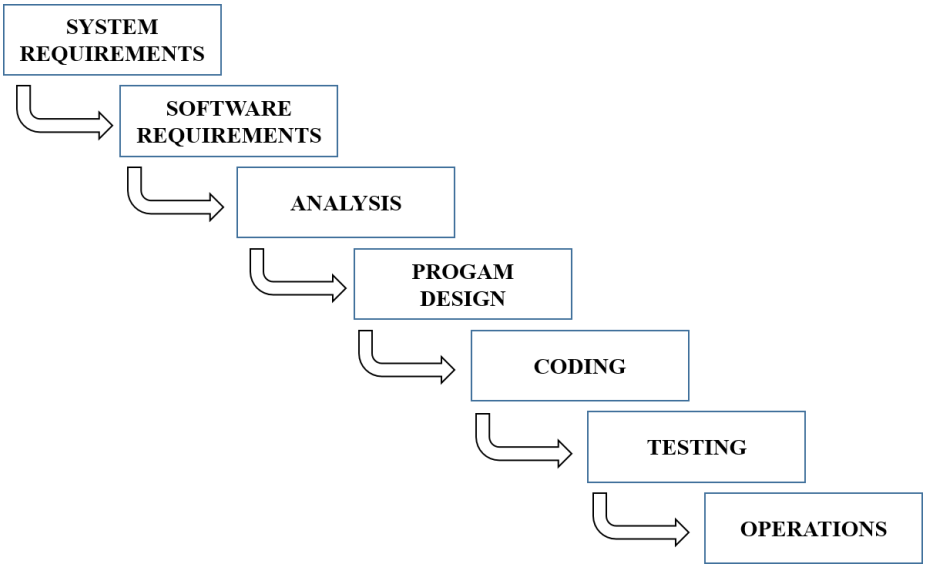

Figure 1: Implementation steps to develop a large computer program for delivery to a customer. (Adopted from [42])

However, this strictly sequential process seems risky and in practice, examples show that there is a step from the testing stage back to the program design, or even to the requirements definition. This is illustrated in the following figure.

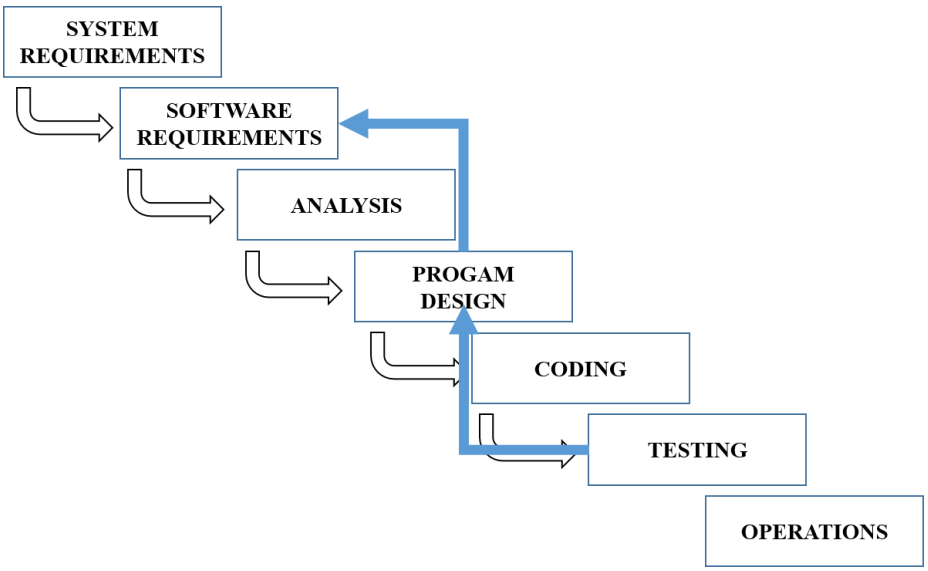

Figure 2: Critical steps and iterations during Waterfall development [42]

The rollback process happens mostly because the testing stage, which is scheduled rather late in the development process can point out changes or differences between the analyzed and eventually experienced solutions or possibilities. In order to fit the violated requirements or designs, the change of the requirements can reach from light modifications up to substantial changes in the design, which is a rather disruptive approach to the Waterfall methodology.

In this case, the process steps would have to be repeated and could cause up to 100 percent postponements in delivery or increases in costs or other efforts. Ideally, a sequential but bidirectional approach is noted to be fundamentally sound, as shown in the following illustration. This almost spiral process is also described as Spiral Development in literature. [30]

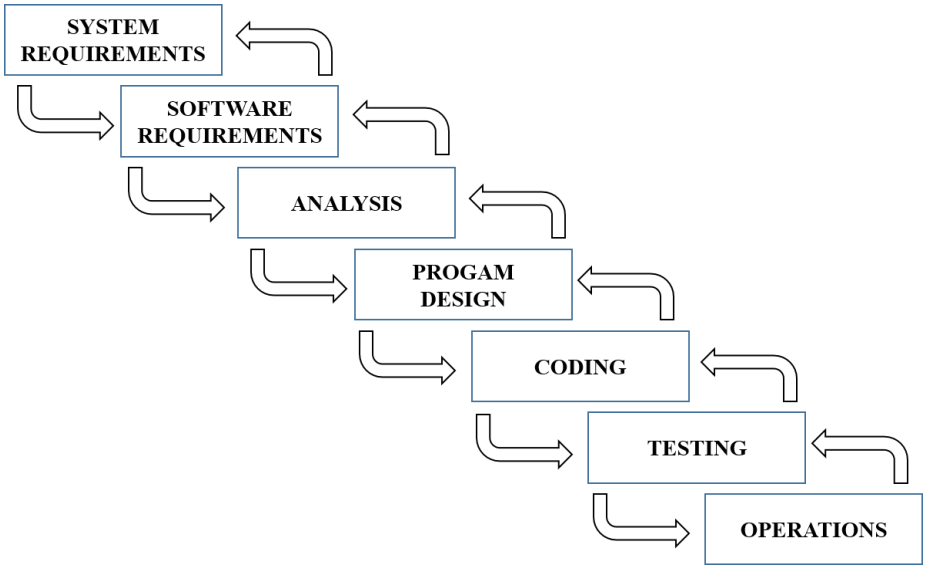

Figure 3: The ideal iterative interaction between phases during Waterfall [42]

According to literature, besides analysis and coding, which are enough to support a simple software development process, five additional features must be added to the basic approach to eliminate most of the development risks. These are described and discussed in the following.

Step 1: Program Design comes first

Before the initial analysis phase, the author inserts a preliminary program design phase, in order to assure that the software will not fail because of technical resources, timing, or data flux reasons. This way, both the program designers and analysts contribute to a meaningful design process which will culminate in the proper allocation of time and technical resources.

This procedure presumes that: 1) the design process is started with program designers, not analysts or programmers; 2) the data processing modes, interfaces and functions are designed, defined and allocated even at the risk of being wrong; 3) an overview document that is understandable, informative and current is written and acknowledged by all members of the team.

Step 2: Document the Design

One of the common characteristics of Waterfall development is the issue of documenting decisions. Certainly, Waterfall development requires more documentation than most

programmers, analysts, or program designers are motivated to put down. However, one of the important rules of managing software development is the *"ruthless enforcement of documentation requirements"* [42].

Various studies and books describe the documentation process as crucial to the holistic software development process. Documentation is often described as an indicator for the monetary value of the code. Undoubtedly, a well-documented code can beneficially support the development process during the testing phase, continuing through operations as well as redesign. [26] states that traditional software development processes i.e. the Waterfall model, are characterized by *"rigorously defined practices, extensive documentation, and detailed planning and management."*

Similar to this approach, [32], who describe software engineering as not only the code and programs but also all associated documentation and configuration data that is required to make the software operate correctly. They further determine the differences between professional and amateur software development by the documentation level. If a program is written for a personal usage, documentation or user guides are optional, but if other users or co-developers are involved, a professional documentation is required. For example, [45] have a simplified view on the Waterfall process, and at the same time the authors put documentation besides requirements gathering and analysis as one of the three most important pillars of Waterfall.

Step 3: Do it twice

Following the documentation procedure, which is the most important process step and success factor, [42] argues that the software delivered to the customer should in fact not be the first finished deployment, but the second version. This step is somehow of importance, since the delivered software should be quality assured and strongly tested before it gets to the customer. Therefore, I would argue that this step is not obligatory or can be solved with the next step.

Step 4: Plan, Control and Monitor Testing

The testing phase is undoubtedly the phase which requires the most manpower, time efforts, management decisions or schedule risks. Testing is particularly a high risk since it is scheduled very late in the development process, and at that stage the alternatives or fallback possibilities are most expensive or impossible. Therefore, instead of repeating the development process twice as discussed in the previous step, it would be beneficial to bring the testing phase forward in the development cycle. The feature of early-stage testing within agile development is a strong advantage over traditional methodologies. Further advantages of agile will be described in the next section, such as the stronger involvement of customers into the production process.

Step 5: Involve the Customer

Undoubtedly, agile methodology stands for collaborative development of software and a highly customer-oriented approach. Yet, at traditional approaches it is not less important to

involve the customer in a formal way already at the beginning, with commitments to features and final delivery. Giving the customer room for changes after the requirements definition will cause trouble and high expenses, in contrast to agile methodology.

2.2.2 Discussion of Waterfall

The prior literature analysis and discussion of the Waterfall model leads to three conclusions which are described in this section.

Traditional or sequential software development can to some extent be aligned to aspects or is based on the process of Project Management. Both approaches show a need for process stages or project streams which will not necessarily contribute to the production of software, but are yet necessary. In terms of project management, this would include project communication, project management or steering tasks, project controlling, or optional project marketing. In terms of software development, the steps which have less or a negative impact on the delivery timing include documentation, decisions or steering rounds, and other organizational aspects. It is one of important tasks of the Project Manager to sell the extended development approach (which includes more than analysis and coding) to both the customer and the development team.

From the time-related aspect, the sequential Waterfall model seems to be beneficial in the first part of the process, whereas the requirements are strongly discussed and analyzed before being set. Also, the fact that system requirements and infrastructural limitations are considered in the program design phase is a plus, if no big changes are expected at a later process stage. However, if some important system constraint or unexpected factor is occurring, the efforts and costs changes within a Waterfall process literally explode and are exponentially rising towards the end of the process. The following table shows a very rough comparison of the Waterfall and agile methodology.

	Waterfall Model	**Agile Methodology**
benefits	In the early stage of the process, Waterfall or traditional, structured development processes are stronger in definitions of requirements, infrastructure, interfaces, and most important – customer expectations and features.	During an initial setup of an Agile process, some notable efforts and time is needed to take the delivery process to an efficient, well-practiced process with different roles and responsibilities.

drawbacks	In a later stage, the costs and efforts to change requirements are exploding. Also, testing and therefore also customer feedback is scheduled only at a very late process stage.	Once the Agile process is set up, the delivery of features and feedback are faster, the customer can be involved into decisions and the complexity of the production seems to be reduced by the sliced product delivery.

Table 1: Rough comparison between Waterfall and Agile Development

Eventually, in waterfall, team roles are sufficiently defined and it is important to clarify the responsibilities among the team members. This is very similar to the agile methodology, since project roles are as important in both approaches. The thesis will further explore the roles and responsibilities among team members and their contribution to the production. Due to limitations in time and complexity for this work, the aspect of how much the team members can contribute or change within the process, open questions will remain after the analysis and can be researched in a further case study.

2.3 Agile Development Methods

In contrast to the traditional software development, agile development methods can be described as more flexible, adaptable and to some extent lightweight models. Compared to the previously discussed Waterfall model, Agile development is not limited to a sequential workflow, but it also allows parallel streams [47] and above all, a much faster processing and delivery.

Most of the agile methods promote iterative development in small increments and have a strong focus on teamwork. A project plan or holistic designs are outlined only at a high level, while the current iteration is going further into details. The teams are working self-organized and decentralized and cover end-to-end functionality [45]. Unlike the traditional approach, the teams equipped with the higher authority, rather than being managed or permanently inspected by a decision board. In an agile environment, progress is measured easily by executable (tested and working) code.

In order to discuss agile development, and as a preparation for the latter case study, this section includes an overview of the common practices, such as Lean Software Development, Scrum, Kanban, Extreme Programming (XP) or Feature-Driven Development (FDD).

2.3.1 Scrum

The term "Scrum", which also has roots in rugby football, where it refers to a *"tight-packed formation of players with their heads down who attempt to gain possession of the ball*" [53], i.e. an "all-at-once" process [47], was introduced to information technology in the late 1990s [43].

The Scrum approach is a general agile method, and it is widely spread across companies in the CEE region. However, Scrum rather focuses on managing iterative development than on specific technical approaches to agile software engineering. Scrum does not include the use of programming practices such as pair programming and Feature-driven (or test-driven) development methods. It can therefore also be used along with a technical agile approach, such as Extreme programming (XP), to provide a management framework for the project.

Referring to Barton [9], the Scrum method stands for a "constant search to simplify complex things" by iteratively reducing the size and complexity, i.e. studying small segments of a large setting and making it simpler through a well understanding.

There are three phases in Scrum [44]. The first stage is an outline planning phase where you establish the general objectives for the project and design the rough software architecture, similar to the Step 2 within Waterfall, which was explained in the previous section.

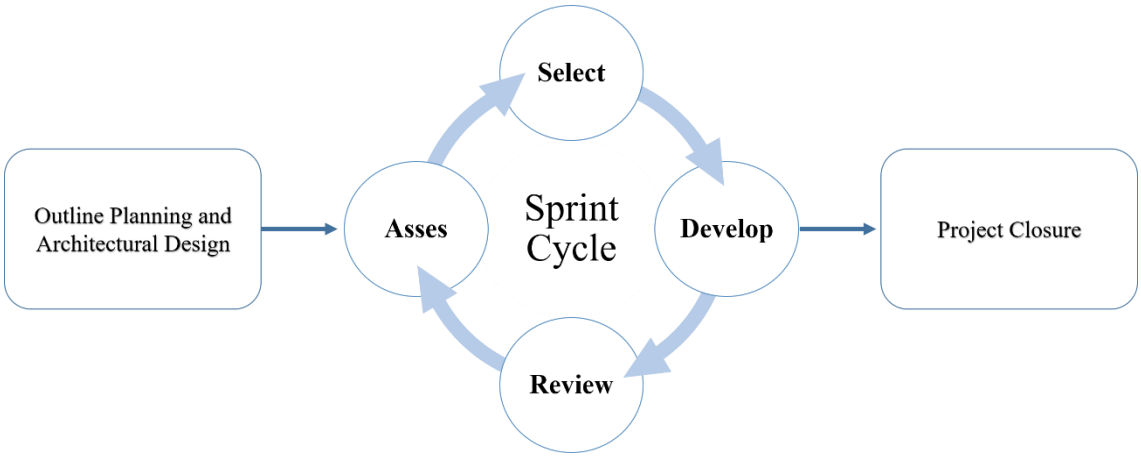

Figure 4: Sprint Cycle (Adopted from [44])

Scrum is an extremely efficient and streamlined process of managing and tracking teams. This happens particularly in the series of incremental sprint cycles. The cycles build the central phase of Scrum, which is also the main differentiator from all other approaches. Eventually, a Scrum project closes with a wrap-up of the project, a completed documentation and user-manuals, as well as a retrospective and lessons learned from the project.

Going into details of the Scrum sprint cycle, it can roughly be described as a planning and delivery unit. Each Sprint consists of a planning of the work to be done, a voting and selection of the features which will be developed within the concurrent Sprint, and finally the development of the committed features.

A Scrum Sprint has following characteristics [44], [45], and [20]:

- Sprints are a fixed length, usually timed between two and four weeks;
- They correspond to a release development in Extreme programming method, which will be described later in this chapter;
- A list of work to be done, the so called “product backlog” is the starting point for each sprint planning;
- A successful sprint planning meeting implies that the product backlog was reviewed, prioritized, the risks and obstacles were assigned.
- The customer is closely involved into the sprint planning and can ask for new requirements or influence the prioritization of the backlog at the beginning of every sprint;
- The review and selection of the features to be implemented involves all project team members, which can also impact the priorities of the features;
- Hence, the selection and voting process is organized democratically in contrast to the organization and decision making approach from the traditional development.
- After the selection and common agreement on the sprint content, the team follows their sprint goals rather self-organized;
- All team members meet on a daily basis for a very short update meeting, a so called “Daily stand-up” (which is limited to around 15 minutes and the participants are usually standing in order to keep the meeting short and focused) to review the progress and reprioritize work if necessary;
- During this stage, the team members are not having alignments with the customer or the rest of the organization, instead the so-called “Scrum master” acts both to support the team and the development process, as well as to protect them and absorb external distractions or requests.
- At the end of a sprint (or at the beginning of the next cycle), the team gets together for a lessons learned or a so-called “Retrospective”. Hereby, it’s not about the features or technical input. The retrospective is a place to discuss the team-related topics rather than technical expertise.
- A separate meeting, the so-called “Sprint review” is open to the customer, stakeholders and the rest of the organization, it includes a presentation and review of the developed features. With a review meeting, the current sprint closes, and the next one starts, until the project scope was accomplished.

As the explained characteristics indicate, the Scrum method allows more than a single Project Manager to take decisions. Unlike a common manager, the Scrum master is rather a *facilitator* [44] who prepares meetings, tracks the backlog, takes notes on decisions, measures the progress, and leads the communication process. Moreover, in the scrum approach, all team members, stakeholders or the customer can take a stronger influence in the development process.

In its original form, Scrum was designed for teams where all team members could get together every day and join the daily stand-up meetings. However, product or software

development increasingly involves teams that are distributed or team members from different locations. Consequently, there is not only one valid Scrum process, but it is also individually modified [25] or developed into a method for distributed development environments – which are also part of the latter case study in this thesis.

2.3.2 Kanban

Besides Scrum, Kanban is one of the widely spread iterative approaches of software development. Besides Lean, Kanban was also introduced in the manufacturing industry in Japan, in the 1950s. Kanban stands for flow control, it was developed to support the just-in-time production [4]. The software industry has been increasingly using Kanban in managing software development projects, as it shows a large potential to increase the interactions between the team members [37].

Similar to the Scrum board, the main characteristic within the Kanban method is the Kanban (i.e. Japanese word for signboard), which is used to document the workflow. According to an empirical study by [37], during their study, they observe that *"with Kanban you are aware of what other people are doing and you can always help them or monitor their work"*. However, the final results of their study reveal that the importance of the Kanban board may actually decrease when the interpersonal communication rises. A very positive benefit is, that this setting seems to foster communication among team members, even if there is no collaboration in between.

Although the effects of Kanban are often analyzed based on practical experience [47] or systematic literature reviews [4], there appears to be a lack of reported scientific research addressing Kanban in the context of software development. In fact, Kanban in software engineering is still a young and unknown topic, as it was only introduced into this context in 2004 by David J. Anderson [6].

Studies show that the benefits of Kanban include customer satisfaction, improved software quality and lead time delivery, an earlier feedback and reduction in customer defect reporting, improved communication between stakeholders as well as increased developer motivation. However, around half of the experiments appear to be experience reports, or are reports at a rather general level.

The literature also revealed that Kanban was often mixed with other agile practices by organizations. But, in order to use such "hybrid" approaches, further efforts on educating the employees as well as the organizational culture were also revealed by a large literature review [4]. Eventually, as there is no unified way of operating a Kanban model, a combination of other agile methods and Kanban is described as beneficial.

Looking towards the case study in this thesis that researches the question of teams and their motivation, the expected results from questionnaires and deeper interviews should contribute

to the empirical research and add further examples of agile methods in practice, or agile versus waterfall (e.g. [46]) to the current research state.

2.3.3 Lean Software Development

Lean software development may also be considered as a rather hybrid approach. In particular, as argued by Barton [9], organizations that have adapted their software development system based on Scrum consider their work as Lean implementation. However, Scrum is considered as Lean not because of the similarity to a lean product development by Takeuchi and Nonaka [49], but because of its adaptive system.

Historically, the origin of lean product development or productions originates from the Japanese industry and is sometimes coined as the "Toyota Production System". In contrast to heavyweight approaches such as Waterfall, Lean is alluding to 'lightweight' and quite opposite to bureaucracy or regulations [45].

In reference to [45], the Japanese approach of Lean production is further considered as a synonym for the increase of productivity, flexibility, speed of turn-around, and quality by continuous improvement and continuous adaptation to a changing environment.

Lean software development is based on the best practices of Lean production, and it was shaped by Tom and Mary Poppendieck [39]. The authors argue that truly lean organizations have a strong competitive advantage because they respond much disciplined and rapidly to market demands, rather than trying to predict the future. Lean software development is all about creating software that is able to adapt to changes in its domain – so to say it provides *"high discipline along with high responsiveness to change"* [40].

The approach involves a set of following guidelines:

- *Eliminate Waste*: Focusing on doing only what adds value for customers and therefore doing it without delays;
- *Amplify Learning*: Using frequent iterations and regular releases to provide feedback;
- *Delay Commitment*: Deciding at the last responsible moment;
- *Deliver Fast*: The maturity of a lean organization can be measured by the speed of responding repeatedly and reliably to customer needs;
- *Empower the Team*: By assembling an expert workforce, providing technical leadership and delegating the responsibility to the workers, the process is decentralized and democratized.
- *Build Integrity in*: Although decentralized and democratized, the process relies on having disciplines in place to assure that a system will integrate and delight customers at any moment.
- *See the Whole*: Using measurements and incentives which allow focusing on the overall goal.

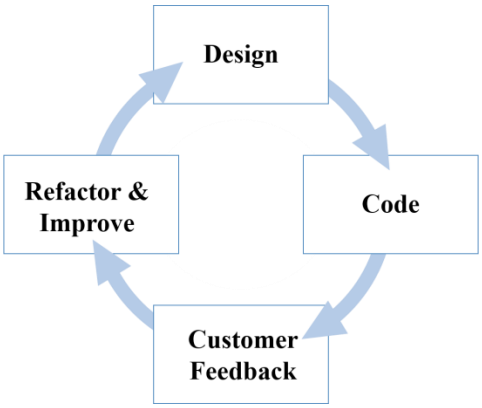

Figure 5: Lean approach to building integrity (Adopted from [45])

Simplifying a complex development process in terms of reducing the complexity of software stacks or an ongoing implementation is extremely difficult and requires a lot of experience in how to slice work into chunks. The idea behind Lean software development is to keep the overall process flexible enough to be able to respond to scope changes.

Even though it is described as lightweight and further away from bureaucracy, the process needs to be organized and monitored. This approach needs powerful tools to support the collaboration and communication within and across teams, and can be supported by Web 2.0-based tools.

2.3.4 Extreme Programming (XP)

Unlike Scrum, Kanban or Lean software development, which focus on the organization of the development process, “eXtreme Programming” (often abbreviated as XP) [10] also implies modeling activities, or the use of specific programming practices, such as a test-first approach.

Similar to a Sprint from the Scrum approach, extreme programming also includes series of releases. A Scrum sprint thereby corresponds to a release development in the XP method. Extreme programming also includes requirements modeling, such as user stories and sketches, it is very explicit about documentation and business decisions [5]. However, it also minimizes modeling efforts by taking a test-first approach to design a requirement in which you develop your tests before you develop your code.

In fact, this may let you think that the approach is similar to the traditional methods, where you plan and think about your software before you actually build it. Because of the stage of understanding what should be built, extreme programming seems to require less modelling efforts. What really distinguishes this development method from the other discussed methods is the fact that developers usually team up into pairs (Pair Programming) as well as particularly the test-before-developing approach. In this sense, it is crucial to clearly understand the customer requirements and transform them into code.

2.3.5 Feature-Driven Development (FDD)

One of agile methods that are based on a strong requirements engineering is the so called Feature-driven Development. The requirements are hereby described in an object-oriented manner, including the term “feature” as the main characteristic [35]. The approach of Feature-Driven Development is defined by three phases: Initiation, Methodology Construction and Termination.

The Initiation phase includes the definition of features, frameworks, classes, and the architecture of the methodology. It is followed by the Construction phase, in which a project manager forms the teams in order to prioritize the development of the features. It seems obvious that this approach is rather similar to the Waterfall method, at which a project manager is empowered to setup and define the team and their tasks. In contrast to that, an agile setting like Scrum is rather autonomous and democratic. Within the final phase of Termination, the product is tested, deployed and maintained. Again, in reference to the traditional methodologies, this final stage, which includes a formal definition and documentation of the used methodology can be described as similar to a Project Definition document as seen in Waterfall.

However, Feature-Driven Development is one of the young agile methodologies. It was initially coined in 1999 by Peter Coad and Jeff De Luca in their work related to Java modeling with UML [19]. The so called Coad Methodology was a predecessor of the Feature-Driven Methodology and is strongly related to UML.

Similar to Scrum, within FDD the features are described in a language which is clear to both the developers and the customers. They are cut into small deliverable tasks that can be divided into short iterations. Hence, multiple related tasks are put together into a work package at the start of an iteration phase, which is scheduled between one and three weeks. After a work package is completed, the results are given to the customers for testing, and the next iteration starts [17].

The next section includes a brief conclusion on the insights from the literature review.

2.4 Conclusions

This section provides a summary of the discussed methods. A first comparison on the traditional versus agile methodology was provided in *Section 2.2.2*. Besides the comparative view that was provided earlier, the remainder of this research will focus on open questions after the literature review, such as:

1) How to describe waterfall and agile software development in practice?
2) Does any of the two methods affect the satisfaction of team members?

3) Does a selection of one of the two methods affect the quality of development?
4) How mature is the development process disregarding the used methodology?

In order to response to the questions above, it is important to point out that the traditional Waterfall sequence is said to be cost-intensive and rather restricted to the sequential workflow of the process. As argued earlier, it is originally based on two main steps, analysis and coding. Later enhanced with further steps, however, the process also includes some steps which are not going to contribute to the final product directly or at all, but because of the larger size and complexity, the development process needs to be administrated. In comparison to this, an agile method seems to be the right method in case of an ongoing delivery of software, constant development requirements or a "sliced" financing model.

The goal of the following case study, which is described in *Chapter 3* is to find answers to the mentioned questions.

3 WATERFALL VERSUS AGILE DEVELOPMENT METHODOLOGY

3.1 Chapter Introduction

In the comparative study two banking corporations are viewed side by side. The companies themselves and the projects done within the companies share many similarities, but also exhibit many differences. The author has directly been involved with both companies, working in projects being compared in the following sections. The conclusions of the case study have been subsequently examined by then-colleagues of the author, but a certain degree of bias is to be inferred.

The companies are not going to be named to avoid possible prejudice to the conclusions, and for legal reasons.

Company A is a multinational banking cooperation with headquarters in Austria. It has branches or subsidiaries, as well as joint ventures, in a number of CEE countries, serving retail and corporate clients, as a full-service bank. By asset size it is placed in the top 100 world banking groups [51].

Company B shares with Company A many characteristics: it is also a multinational banking corporation, with a strong focus on CEE countries (e.g. Czech Republic, Slovakia, Serbia, Slovenia, Croatia etc.), serves all clients ranging from retail to large corporate clients, and is a full-service banking group. Some of the differentiating characteristics include: wider market presence (including Turkey, Poland, Russia, Ukraine, etc.), higher total assets (about four times) [51] and higher relative position in the European sector.

Business and market specifics are not taken as a factor in this survey, as it was deemed that they have little or no impact to the everyday work of the delivery team member.

3.2 Company A: Agile / Scrum

Project examined in the Company A was part of a wider initiative within the area of medium to larger corporate clients, which was regarded at the time to be within the overall strategy of the corporation framed within the statement: "We want to be the first bank of every medium company in the country".

The effort was organized not simply as a project, or a group of projects, but within the new organizational structure, with a clear hierarchy. Every project within this structure was initiated by the department head, through direct contacts with potential external clients (large corporate clients) or internal customers (e.g. key account managers). After a certain period of inception, the ideas were formed into vaguely described products, the staffing of the

department was initiated, and the development methodology was chosen to be Scrum, using Java programming language, as is common within the company.

Majority of people hired internally were former project managers, with significant knowledge and experience in banking industry. The selection was done as a mix of internal and external employees in order to make so called "innovation factory", and mostly given the task of product owners and scrum masters. Several external developers were hired on a limited contract to join the team in their development process. In total, at its prime, the extended agile team had about ten people, including facilitators, testers, business partners, team leads, etc.

Team set up was done with extensive support of an external consulting company, which helped setup trainings in Agile methodologies, specifically Scrum. The consultants had significant experience with implementation of Scrum in numerous companies, including the ones that had remote working places, and those that had very complex projects with many teams, being synchronized through "Scrum of scrums". Training was also attended by mid and higher management, as well as business counterparts of the expected projects. The general feeling after the trainings was overtly positive, and enthusiastic. The support of the external company extended into the implementation (helping with scrums, weekly retrospectives), and testing.

In retrospective, this approach has all the characteristics of a well thought and well executed transformation from a general waterfall approach, dominant in the company, to an agile methodology of scrum.

However, the external environment has changed significantly within the frame of a few months. This change is summarized in a changed credo of the top management "*We want, as has always been in the past, to remain the best retail bank in our markets*".

This change has been gradual, but nonetheless disruptive: the financing was cut, and changed from lump-sum to iterative, gate-based process (similar to stage-gate model, Cooper), where the team (or management) had to go periodically for approval of additional funds. Additionally, the original scope of 4-5 projects/products was cut to one, with all the others put on best-effort scenario, e.g. minimum external costs can be involved, and/or other departments have to be bought-in to become investors.

External environment was also the cause of another perceived problem: a limited involvement of internal clients. Due to budgetary limitations the project was forced to widen the scope of the product from larger corporates to small and medium enterprises as well. With such distributed ownership often some divergent ideas happen, which can ultimately threaten the delivery of the project. Taken into consideration the threat to the future of the department, the scrum master and the product owner, both internal employees of the department started more often to compete for the dominant role in running the relationship with the business.

During the implementation phase budgetary problems became more obvious, and this put more pressure on the external team members. Internal members grew also restless because no

additional projects could be started, so they were increasingly outsourced to different departments, for running traditional waterfall projects.

In the end the product was launched, but no subsequent data can be acquired on whether it is used and the user satisfaction. Out of the starting 10 team members, 2 remain. This brings us to conclusion, that even the best structured agile transformation processes will remain unsuccessful if not reflecting the general strategy of the company.

3.3 Company B: Agile turned waterfall

Company B started a significant, multi-year, multi-million-euro project of revamping the whole customer-facing banking portal, including public website, internet and mobile banking, as well as building a brand new development department from scratch, in one of CEE countries.

Although the aforementioned components are closely interweaved, the same content being delivered on all channels through Omni-channel approach, and using the same Service-oriented architecture (SOA), in this case study for clarity and simplicity we focus only on Internet-banking component.

Project kick-off was in August 2014, starting with the scope. At this point the time dimension of the project was set to 10 months, from inception to delivery, with top management influencing heavily this decision. Starting with this target date all the planning, resource allocation and infrastructure scoping was reverse-engineered in order to meet the deadline.

Starting with the scope, and cost, the requests for proposal were requested from the partners of Company B, both for technology stack, and the actual implementation. Two different partners were selected, first partner (from now on: Framework partner) for development framework, and second partner (from now on: Development partner) for implementation, and the development department was to be placed into one of the CEE countries with booming IT scene.

Development partner at the time of winning the tender did not have any employees in the chosen country, and service-oriented architecture was not present. After the initial hiring those early teams started working on mockups and we formed an opinion that these prototypes were not properly communicated to the management, resulting in unrealistic expectations. One of the interviewees said: "*What they did then can hardly be called development*".

Government of the project was given to Development partner, with their project managers and team leads. They also employed the large number of developers, bringing them to about 50 at their peak. Company B internally and externally employed an additional about 20 of (internal and external) team leads, business analysts, scrum masters, project managers, project management office, testers, management, etc.

The selected family of methodologies was agile, specifically Scrum, but no real, in-depth training was given, and no supervision was present to ensure adhering to agile principles, resulting with each team within the development department practically running its own scrum-like approach.

Supervision was kept internally to Company B, but without hands-on approach, the Company B supervision had to rely on feedback they received from Development partner.

Approximately 6 months in the project, in February 2015, the second office was setup at a different location, and first employees were hired for this office, with some employees being transferred or commuting between the offices.

In March 2015, first services were exposed on Enterprise service bus (ESB), being the backbone of the SOA architecture, and for the first time allowing end-to-end integration. Product-owner (PO) group got extended at this time, in an attempt to bridge the apparent problem of understanding the requirements of the project.

In June 2015, the project was placed in User acceptance test stage, and some 1500 defects were recorded, making it apparent that the deadlines have not been met and that there was some lack of understanding from the management on the stage and maturity of the project. At this time the developers were grouped into two streams: fixing stream and evolution stream, making the complexity even harder to manage.

At this time sprints, with sprint planning, daily stand-ups and retrospectives are taking place, but the general problem was reported as being "*The seniority of the factory was really low, making it difficult to build self-managing teams*".

Development partner at this time pointed out the problems were caused by the changing technology stack from Framework partner, but in general this was not held enough to excuse for Development partner's poor management of the project.

Therefore, in August 2015, the project management and team leads were internalized, keeping, for the time being, developers employed with Development partner.

In September 2015, all the distributed teams were dismissed and the single office was kept. The development was almost fully internalized, with Framework partner now being in charge of technical leading and refactoring. Teams were reshuffled trying to distribute seniority in most efficient way. Scrum was used for new development, while Lean software development was employed for bug fixing.

By November 2015, several deadlines were broken, leading to final departure from Scrum (December 2015), reverting gradually to waterfall methodology. Delivery is done at the end of each day. In a morale boosting attempt the "end of development" was announced, even though it was clear that not all the necessary features are ready to be shipped or haven't even started to be developed.

Next "go-live" date was scheduled for January 17, 2016, at which time there were still 500 unresolved bugs. By April 2016, number of bugs is reduced to about 100.

At this point the development is done through an interesting combination of methodologies "*a mix of everything, maybe best called Defect driven development"*:

- All the new development is done in sprints of 3-4 weeks which result in code being delivered to system-integration environment
- Bug fixing on this code is done through lean software development which is concluded by the unit being published on user-acceptance environment
- After bug fixing the unit is deployed to production environment
- There are no scrum masters, no sprint planning, but the role of product owner is kept
- Tasks are allocated to team members by the respective team leader who then tracks the statuses

It is currently unclear about the consequences of such changes in methodologies before the product is fully shipped. While [46] conclude that it is not completely uncommon that companies fallback to waterfall when they see that their current approach is failing this approach was not fully taken because it was concluded that no full waterfall can be achieved at this stage.

At the time of writing this thesis the product is in pilot phase and is yet to be shipped to full-range external clients, as the customer (i.e. business side of the corporation) refuses to accept the product which is performing worse in speed or quality. Those issues still remain to be resolved.

This case study brings us to conclusion that even the projects which are fully aligned with the corporate strategy, provided with enough funds and other resources, can struggle to meet the goals. Principal causes can be found to be:

- arbitrary set deadlines by the management,
- insufficient monitoring of the process,
- changing development framework,
- junior developers,
- changing organizational structure,
- changing employment,
- big fluctuation of employees,
- lack of experience of the management,
- postponing the decisions until it was too late,
- not adhering to the principles of agile.

Given such history, it is not to be expected that anything will change for the better when the methodology itself changes.

One of the working theories when approaching this company and this project was that IT projects will suffer from the environment more than internally. This was to be expected because Banking and financial industry came to be regarded as rigid and resisting the change, but from our interviews and personal experience with Company B and its partners we came to

conclusion that most of the reasons for poor performances are coming from internal reasons, listed above.

3.4 Summary of Practice

In the preceding sections we looked at two examples from the banking industry, with very different environments and experiences encountered. While in the Company A it seemed that everything was set up perfectly for success, the environmental circumstance prevailed and in the end the project was abandoned. The outcome of the rather innovative initiative, a fully developed application – even though functional and integrated with company's systems – was rarely used in practice after development.

In Company B, the methodologies, structures, financial and other agreements were continually changing, but the environment stayed focused on the necessity of the product development, and its quality and functionality. In this case the project seems endangered with its internal problems, and not the external issues.

As a result of these two short studies, we can conclude that while proper application of a methodology (agile or otherwise) is not a guarantee of a successful project (Company A), its absence or instability (Company B) can attribute to additional difficulties in otherwise anyways sensitive process of software development.

4 CASE STUDY AND RESULTS

Following the literature review in *Chapter 2* and the discussion of Waterfall versus Agile methods in practice in *Chapter 3*, this chapter focuses on the hands-on research process behind this thesis. The chapter is organized as following: an introduction into the derived research questions in *Section 4.1*; the used research tools in *Section 4.2*; explanation of the data collection and analysis in *Section 4.3*; the main study results in *Section 4.4*, and eventually conclusions on the conducted research in *Section 4.5*.

4.1 Research Questions

Besides the introductory theoretical comparison of the two software development approaches in *Chapter 2* and the insights into practical cases of both methods in *Chapter 3*, this section focuses on answering the following research questions in a deeper sense:

1) Is there a singular recipe for introducing agile software development in corporations?
2) Which factors affect the satisfaction of team members?
3) Which factors affect the quality of development in both methods?
4) How mature is the development process disregarding the used methodology?

Eventually, in order to develop basic conclusions for corporations on the one side, and contribute to academia on the other side, this research work focuses on answering the research questions after insights into theory and literature on software development, enhance the insights with remaining open or unanswered questions and compare the theory to practical examples of software development

4.2 Research Tools

For the purposes of this questionnaire a relatively simple tool was required, having the following characteristics:

- Online tool, to provide for territorial distance and time difference between the participants
- Supporting principal question types: e.g. multiple choice, short answer, long answer, Likert scale, mandatory and optional answers, and basic presentation/exclusion logic
- Data export and basic reporting

Having in mind the abovementioned requirements, as a principal data collection tool we selected online platform – *SurveyGizmo*, primarily for the reasons of speed and flexibility of

building an online survey. Since the survey was by invitation only and we didn't require any marketing or tracking tools we selected the Basic plan.

Later in the process, when extracting and analyzing the details of the survey, we used Microsoft Excel to compare quantitative data and provide visual survey insights.

4.3 Data Collection and Analysis

The data was collected through the above mentioned research tools during the period between June and August 2016. In terms of time-dependency, it is not crucial for the case study at what moment the data was collected, as the case study focuses on the impact of a particular methodology on the final product development. In case of a rather quantitative analysis of the two methodologies, such as the questions, which approach is more effective, i.e. produces less bugs or proves a faster bug-fixing process, the time variable would be crucial.

We applied the previously used framework or questionnaire by [7]. However; the questions were adapted to the examined cases. In particular, the questions concerning the appreciation or recognition of team members were pointed out to understand the motivation of team members in their particular role. The adapted questions from the original study [7] was used to question the maturity of agile development methodology, which was also applied within this research work. An additional questionnaire part was added to the initial survey, in order to examine the employees' satisfaction, as well as the appreciation of a specific methodology.

We used an online survey to collect data from team members from both Case A and Case B. The developed questionnaire contains multiple-choice questions based on Likert scale, as well as open-ended questions. The team members' participation to the study was optional and there were no preselection criteria on who should participate in the study. Also, the participants could choose to remain anonymous during the study, which was used rarely. In general, we found all roles among the respondents to the survey, so it can be said that the survey results are broad and holistic. Due to the limitations within this research work, the survey was not decoded based on gender, age, etc. The focus of this study was a comparative view on the two cases, rather than insights into demographic data.

SurveyGizmo, the selected research tool offers a variety of visualizations of the questionnaire results and therefore also allows interesting insights into results of the study. For example, we created multiple visualizations of selected questions and thereby compared Company A to Company B and to the overall result. Our experience with the tool has shown that it is very supportive during empirical analysis, including the setup, monitoring and evaluation of the questionnaires. The next section provides a detailed analysis on several research questions.

4.4 Main Results

The survey was filled out 12 times, with 6 respondents from the Company A, and 6 from Company B. This was within the expected range, as the invitations were targeted to relevant team members that also expressed the readiness to participate in post-survey interviews. Out of all the participants who started the questionnaire very high 80% finished all the mandatory questions. Remaining 20% were taken into consideration when observing the aggregated results, but were excluded from comparison between the companies. In the following sections main results are examined.

4.4.1 Demographics

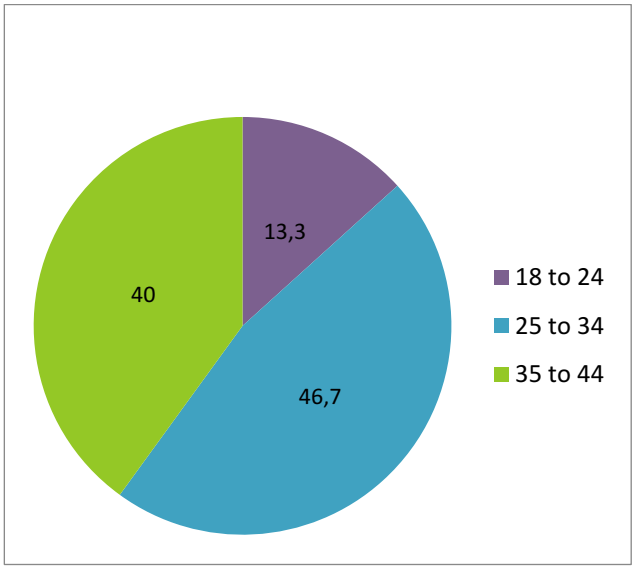

Figure 6: Age distribution of the respondents

Figure 5 shows overall distribution of participant ages, with similar number in the age group of 35-44 and 25-34, with a remaining 13% below 25. It is also interesting to compare the Company A and Company B with regards to age, as shown in Figure 6.

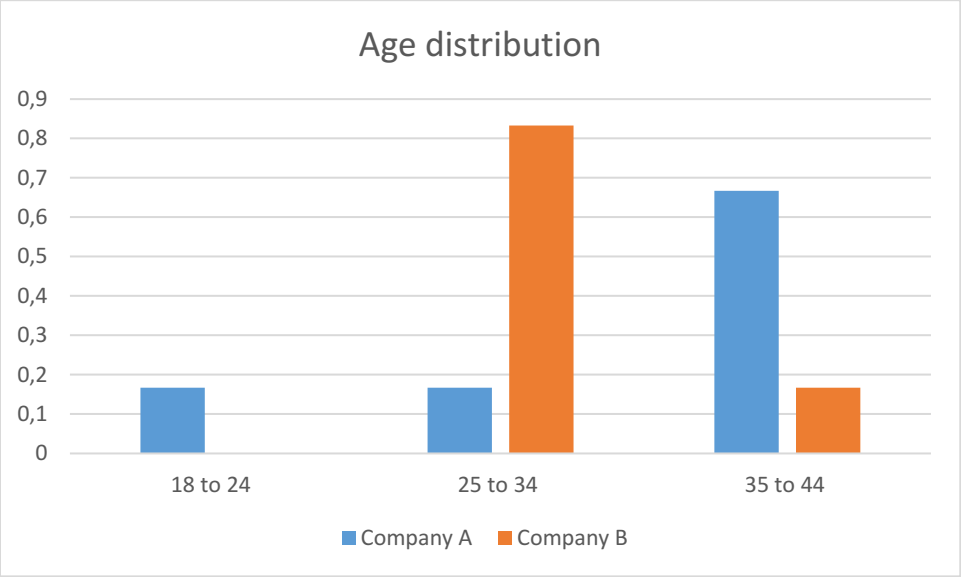

Figure 7: Age distribution comparison between Company A and Company B

It is interesting to see that Company A has exactly the same proportion of people in the range of 25-34 as Company B has in the 35-44 range, and almost the same result is for the proportions in 25-34 and 35-44 respectively. This discrepancy can be explained by the newly formed team in the completely new department in Company B.

Education structure is the same for both companies, and highest level is in both cases 60% Master (or similar) degree, and 40% Bachelor, as can be seen in Figure 7.

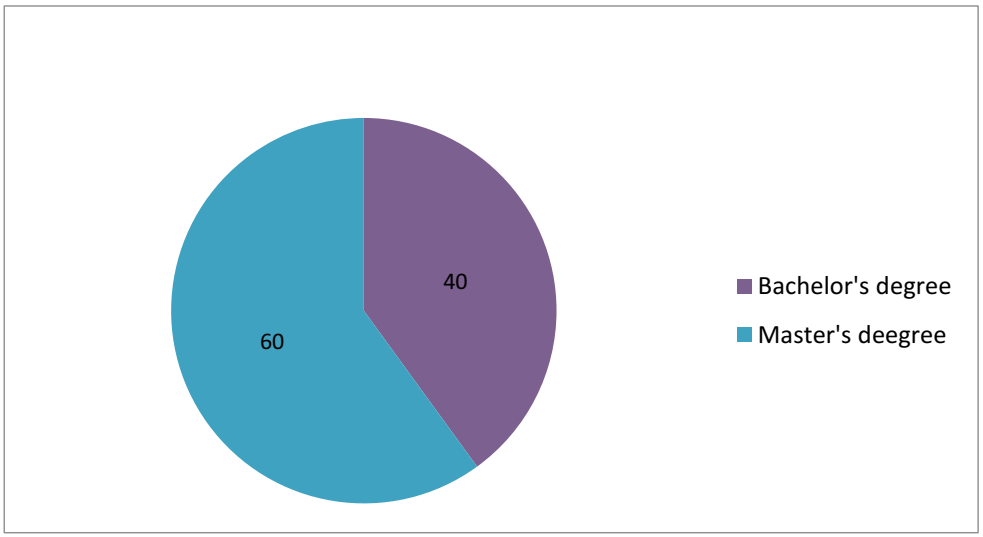

Figure 8: Highest level of education

Another data corroborating the aforementioned picture of Company B running a startup-like project is the comparison of years of company affiliation (Figure 8).

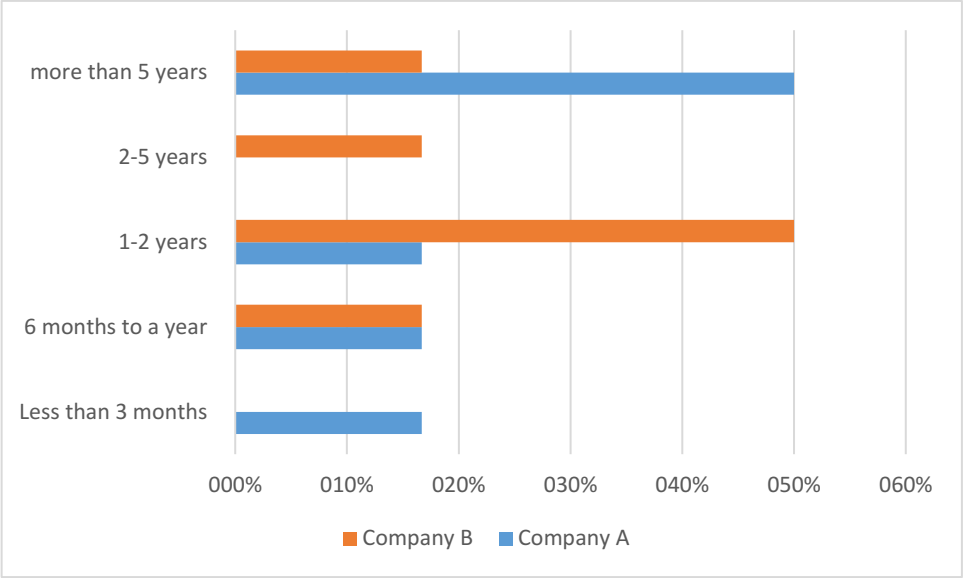

Figure 9: Duration of company affiliation

In Figure 8 we can again observe that majority of respondents in Company A have more than 5 years of affiliation, while the respondents from Company B have 1-2 years, which means that they were hired specifically for the project, and have no prior affiliation with the company.

It is notable that all the participants state that they have experience in agile methodologies, and most state experience in waterfall, as can be seen in Figure 9.

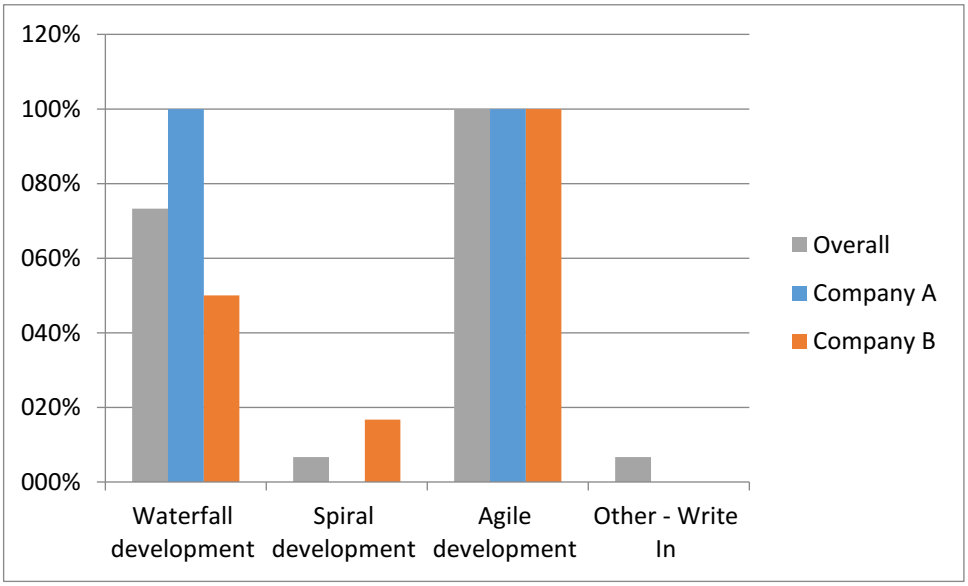

Figure 10: Familiarity of technologies

When comparing both companies, the biggest difference is that all the participants from Company B state that they are familiar with Waterfall development, compared to only 50% from Company A. This also confirms the observances about the methodology uncertainties and methodology switching in Company B.

4.4.2 General employee satisfaction

In this section of the questionnaire we tried measuring overall satisfaction of the employees/contractors with the company in question. Given the number of respondents it would be statistically irrelevant trying to correlate to methodologies in a relevant company. Therefore, the main goal was to get participants thinking about their job satisfaction prior to discussing the employed methodologies and to provide context information for planned personal and targeted interviews.

When asked whether they feel valued as employees (or contractors) the responses in both companies were identical: 16,7% strongly agreed, 50% agreed, 33,3% neither agreed nor disagreed, but counting in the anonymous responses, there is a drop in “agree” and “strongly agree and a rise in “strongly disagree”, as seen in aggregated chart on Figure 10.

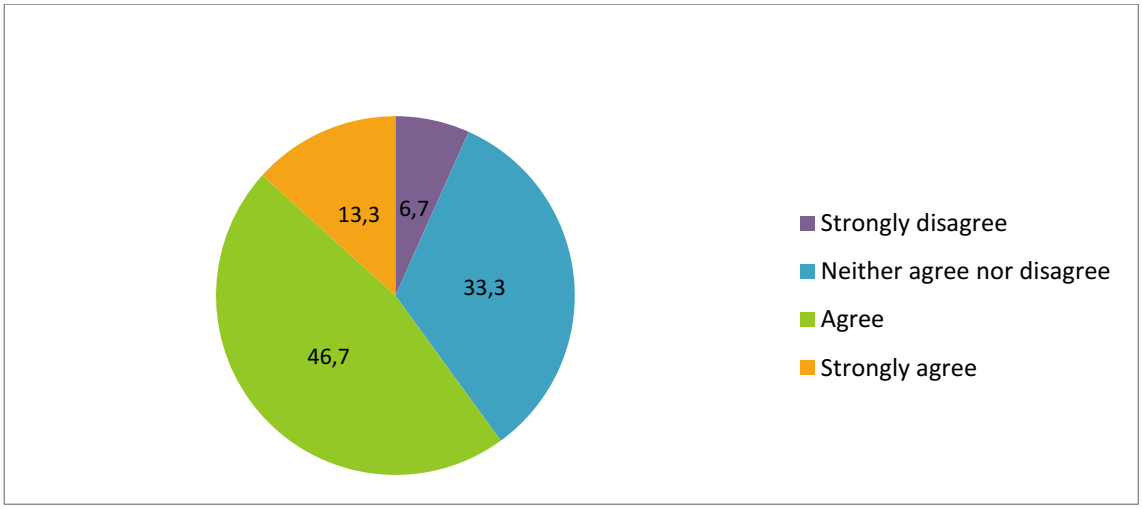

Figure 11: Responses to the question "I feel valued as an employee"

Even though the number of responses are too few to form firm conclusions, we would suggest a theory that the really disgruntled employees feel reluctant to share their dissatisfaction in fear of provoking retaliation in the job environment, even though it would be really interesting for this or future studies to investigate the sources of this dissatisfaction.

When looking into job requirements, in the case of Company A, the distribution is more evened on the scale, and more on the positive side, having 50% agree, or strongly agree, as compared to 16,7% in Company B. This can be seen on Figure 11.

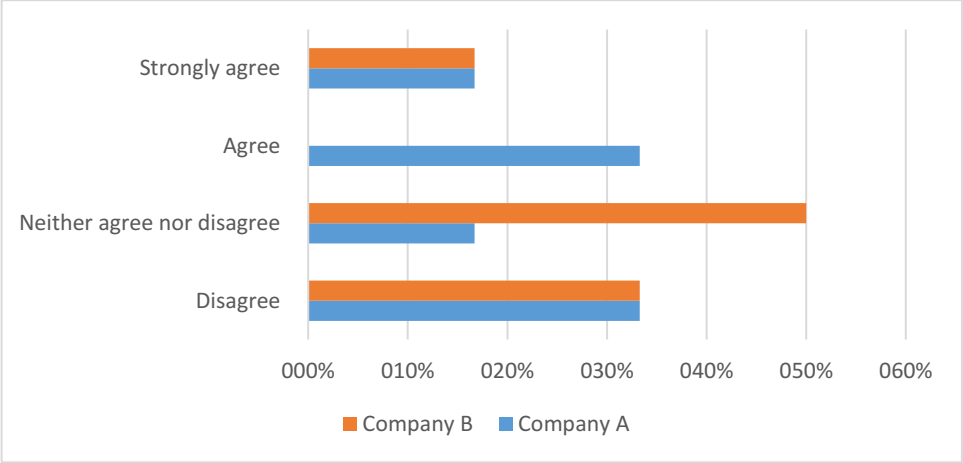

Figure 12: Clarity of job requirements comparison

Even larger contrast appears in the following question: "I receive the training I need to do my job well", where Company A has 66% of responses in Agree or Strongly agree, while again only 16,7% of Company B employees feel the same, as shown in Figure 12.

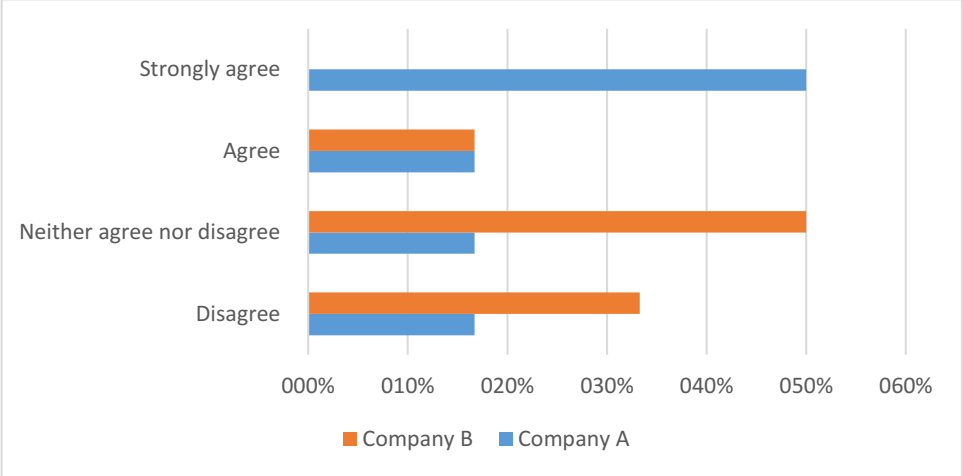

Figure 13: Adequacy of job training

All the rest of the questions asked in the area of employee satisfaction can be examined in Figure 13.

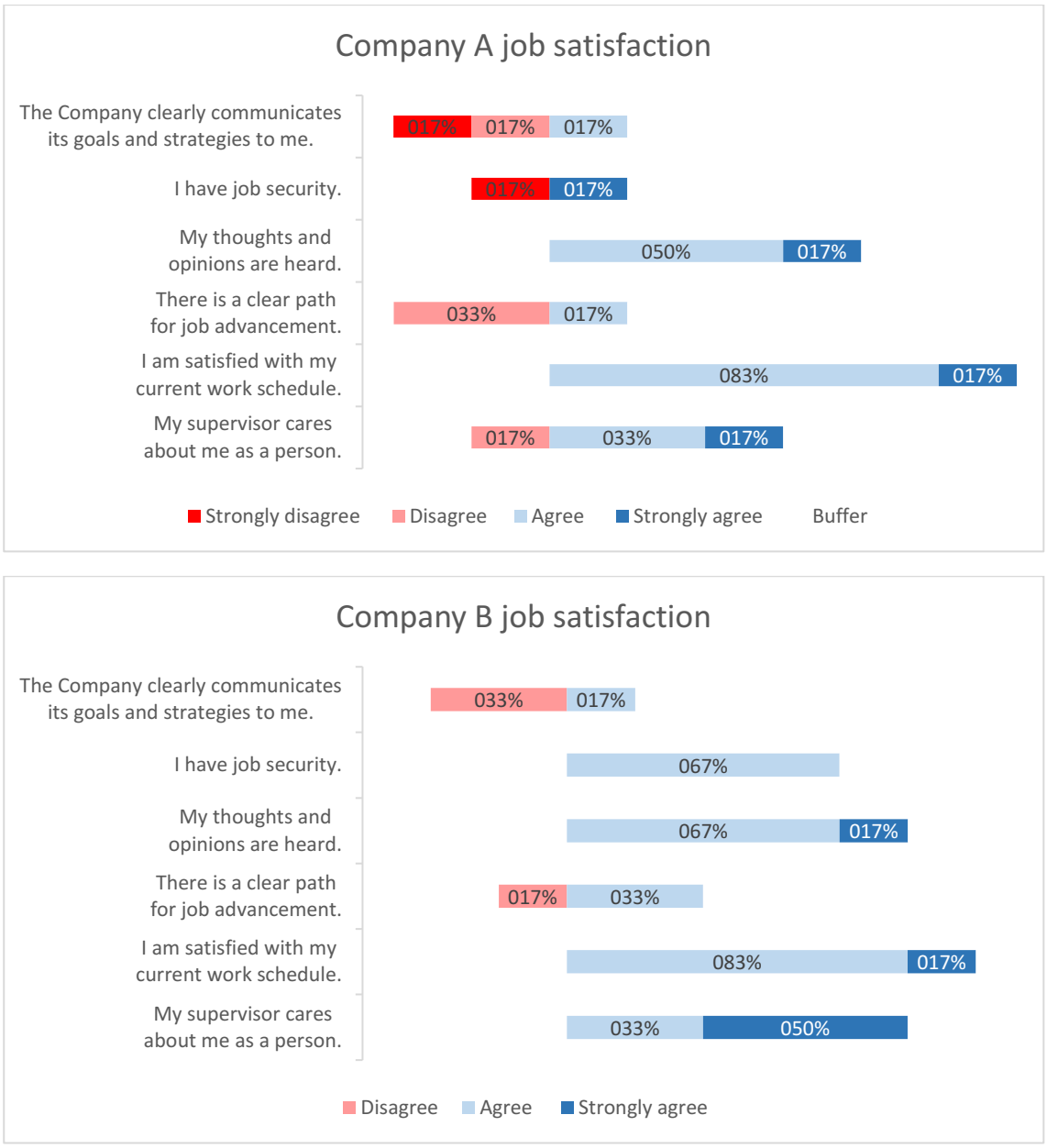

Figure 14: Job satisfaction comparison between Companies A and B

From figure 13 we removed the neutral positions in order to highlight the leaning positions, remaining answers show that while in both companies respondents are satisfied with their work schedules, on most other questions Company A satisfaction rates lower. The most obvious discrepancy is shown in the "job advancement" question, where Company B positive responses amount to double of negative, and exactly the opposite can be seen in Company A.

There can be multiple explanations for this: perhaps the maturity of the team and relatively more rigid structure in Company A also accounts for lower vertical mobility. It is also observed that, on average, respondents from Company A have been working for the company longer (see Figure 8), and are of higher average age (see Figure 6). While this research cannot

provide answers for this due to scope and time restrictions it would be an interesting topic for future research.

Full scope of answers, including the neutral responses, and additionally the survey postings for which no company affiliation could be determined (e.g. partial, anonymous, etc.) can be seen in Figure 14.

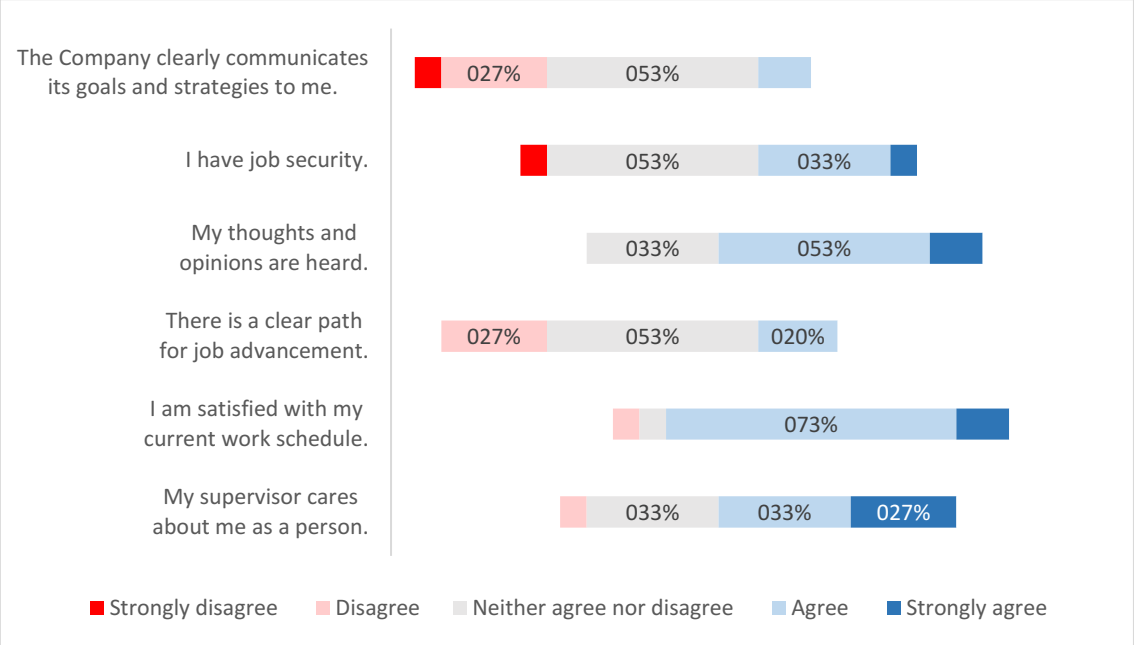

Figure 15: Job satisfaction across all the respondents

Numbers depicted in Figure 14 don't differ greatly from those previously discussed, even though by including anonymous replies we see a negative rise in a category which was previously completely positive: "work schedule". This is in line with prediction that employees with a negative experience with the company will avoid disclosing it within the survey.

4.4.3 Waterfall development experiences

In this section (and contained subsections) we explored both the maturity of the waterfall methodology in each of the companies, and the personal experience of waterfall development within the respective company.

Our respondents have been filling a variety of roles, which can be due to very flexible environment (or chaotic) or due to the roles being poorly defined on the respective company's level. The reported spread of roles can be seen in Figure 15.

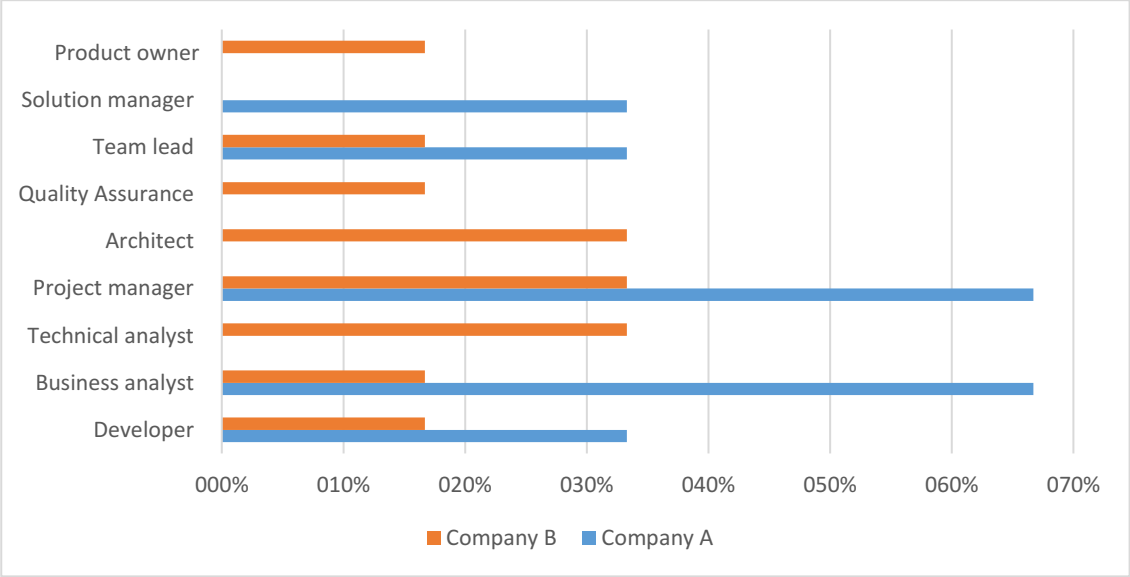

Figure 16: Distribution of roles in waterfall methodology within the companies

While usually Waterfall doesn't define the role of a Product owner, one of the respondents reported this role in the freely editable answer field. This is interesting, and good topic for the post-survey interviews. When comparing the respondents by company we observed that employees of Company A reported 5 roles, while the employees of Company B reported 8, including the freely editable entry.

Maturity of waterfall process was measured by statements reflecting the target properties which a waterfall methodology should have: well developed project plan, formal process, etc. Each of the offered responses has an associated value, starting with 0 in case of complete non-conformity to the principle, 1 when partially conforming, and 2 when fully conforming. Several statements could have in some cases the same values, as it was perceived to bring the same benefit to the maturity of the process. Results of this sections can be seen in Figure 16.

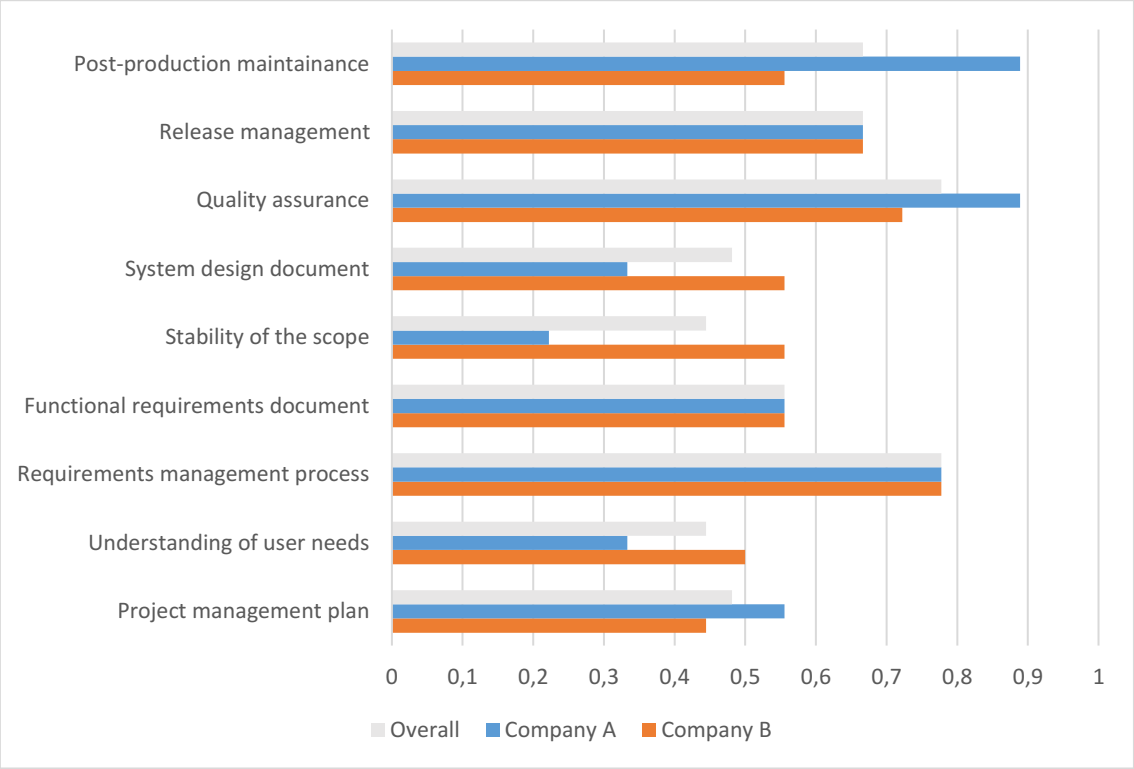

Figure 17: Maturity of agile methodology components (higher is better)

As can be observed in Figure 16 "Requirement management" (including a formal sign-off) and "Quality assurance" are regarded as the strongest points of the waterfall implementation in both companies. On the other hand, the weakest points of overall waterfall implementation are "Understanding of user needs" (and their analysis) and "Stability of the scope". In both of these weak points the perception is that components are significantly worse in Company A in comparison to Company B. One area where perception in Company A is significantly better is "Post-production maintenance", probably due to maturity of the environment, not necessarily the methodology.

4.4.4 Perceived satisfaction during waterfall projects

In this section of the questionnaire we attempted to measure perceived satisfaction with the company and the team within the context of waterfall methodology. The block consisted of 6 questions using Likert scale. If each option was valued from minimum 1 points for highest dissatisfaction to maximum 5 points for highest satisfaction, then all participants' answers across all questions show 63,77% of maximum possible outcome.

Satisfaction perception per question is shown in Figure 17.

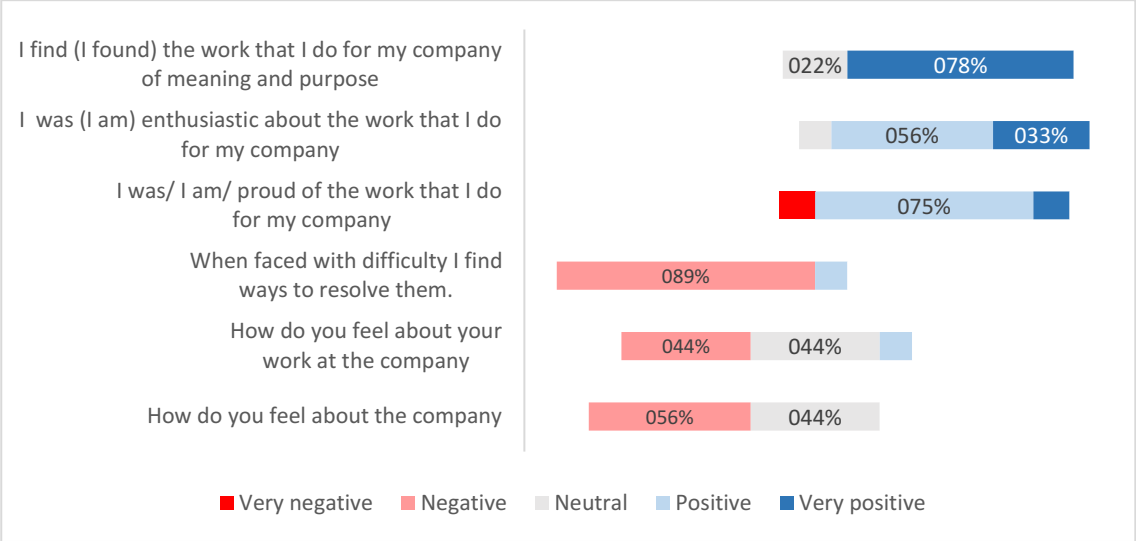

Figure 18: Satisfaction while working in waterfall methodologies

What we find interesting about Figure 17 is how the answers shifted from mostly negative (the first question asked is shown in the bottom of the graph) to mostly positive, and it is also pretty interesting to observe the abrupt switch. From "When faced with difficulty I find ways to resolve them" to which most respondents (88,90%) answered "With some difficulty", to the very next question "I am proud of the work I do for my company" 87,5% answered either "Somewhat" or "Very enthusiastic".

4.4.5 Agile development experiences

Agile development block of questionnaire followed the main principle of waterfall block: general characteristics, maturity and satisfaction. In this and following two sections we will present and interpret the corresponding data. Our correspondents were first asked to provide some background information on their roles within the respective company, the results can be seen in Figure 18.

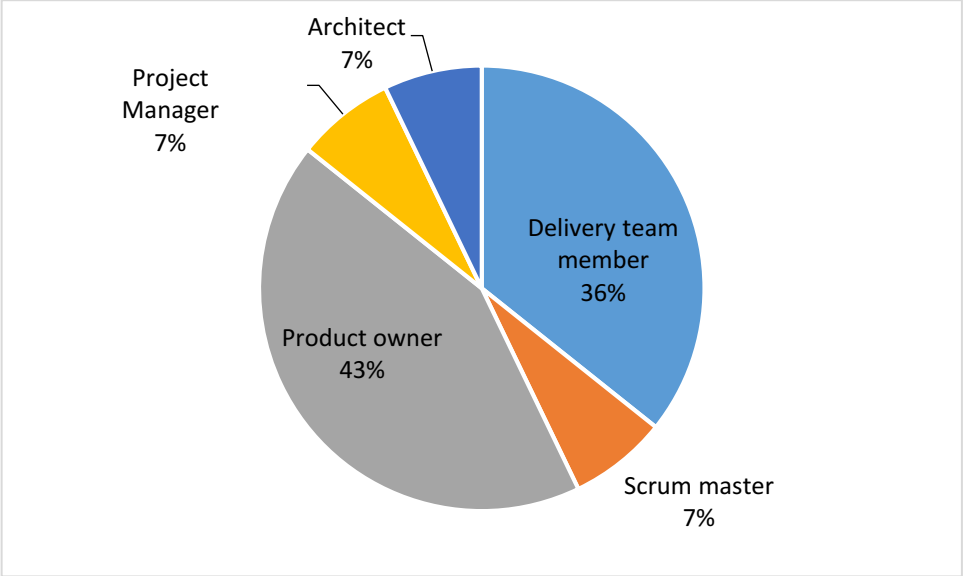

Figure 19: Roles performed within agile methodology practice

Those participants who reported being a part of a delivery team had one or more of the following roles (each role having equal frequency of occurrence):

- Development
- Testing (QA)
- Specialist
- Process Coordinator
- Kanban Master
- Team Lead
- Project manager

Following three groups questions were designed to measure the maturity of the waterfall process. Having in mind a relatively high number of questions in this group (37) instead of the Likert scale we used Yes/No questions asking the participants to select "Yes" if they mostly agree, or "No" if they mostly disagree. While the idea behind this block was to have everyone select either option, we allowed questions to be skipped.

For the purpose of interpretation of agile methodology maturity, we assigned (1) point to "Yes" answers and (0) to "No". Graphical view of the results can be seen on Figure 19.

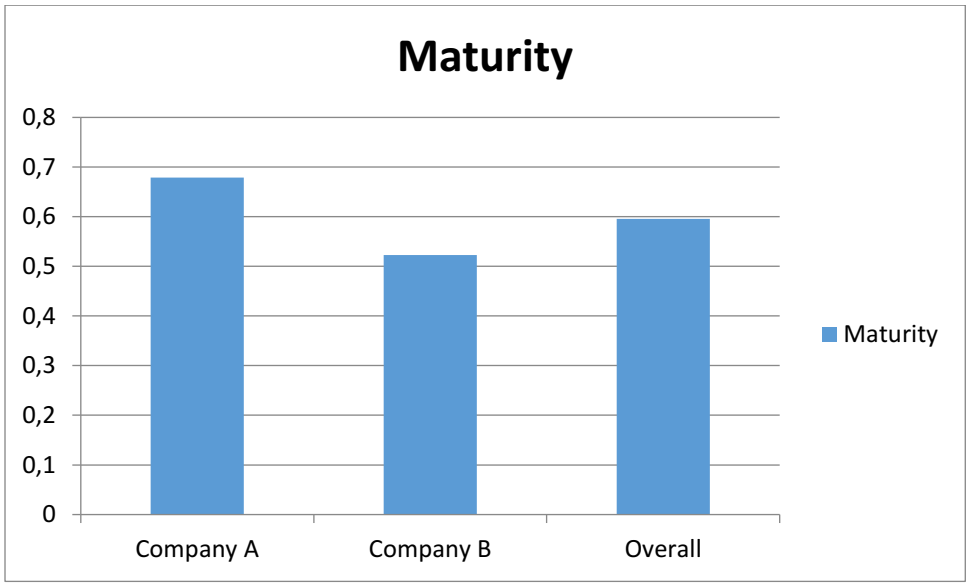

Figure 20: Relative maturity level

Reported characteristics of agile in Company B suggest lower maturity than in Company A, which was expected due to more thorough education strategy and more structured approach of Company A to Agile methodology (as described in chapter 3).

When looking into individual questions, in a large number of answers Company A showed significant positive difference. If we single out only the questions where the maturity index (percentage of „Yes“ answers in all answers to a question) difference in favor of Company A is more than 40% we get this list:

- If used, hardening (or stabilization) iterations are scheduled in advance (+42%)
- There is a clear and common understood definition of done/acceptance tests for completed features (+43%)
- Dependencies are well-managed (+50%)
- End-of-iteration demos occur (+50%)
- Unit testing occurs (+50%)
- Stories are estimated in points (+63%)
- Stories (user requirements) are written in a way that describes how a user can benefit from the feature (+67%)
- Retrospectives occur periodically (+67%)

Number of points where Company B is more mature is significantly shorter, even when the difference threshold is lowered to 20% it yields only 7 questions:

- Scope of each release is planned in advance (+20%)
- Team members work on finishing each iteration as a team, helping each other along the way as needed (+25)
- Team members pair-program at appropriate times (+25%)

- Product management or someone in that role is integrally involved (+27%)
- Iterations are loaded to the right capacity (+33%)
- Code reviews occur (+63%)

From the comparison of major advantages between Company A and Company B we conclude the relative strengths of Company A are in the requirements and process management, while Company B is stronger in the technical and practical aspects of software development (e.g. code review, pair programming, team work).

4.4.6 Perceived satisfaction during agile projects

As part of the survey we measured the perceived satisfaction of the participants who reported having experience with agile methodologies in the company relevant for the survey. Findings are shown in Figure 20.

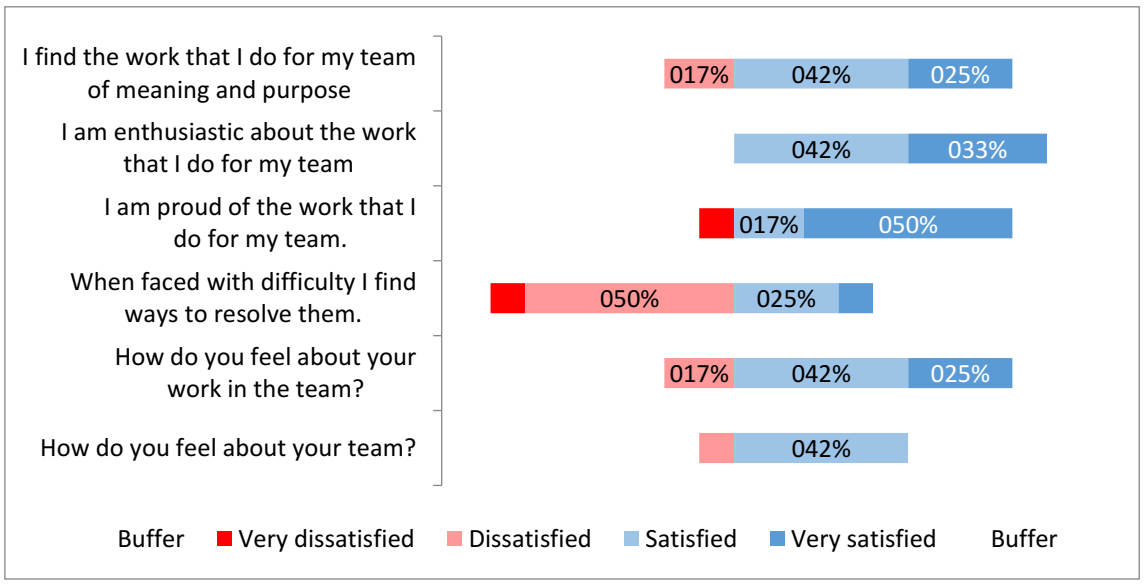

Figure 21: Perceived satisfaction in Agile projects – combined data for both companies

In Figure 20 the neutral answers have been removed for easier readability of the trend across answers. Interestingly the most neutral answers were present in the question "How do you feel about your team" which might mean that the employees in general are cautious when expression opinions about others, especially when it might be perceived as a personal and not professional opinion.

Resolving job related problems present a difficulty a significant 58% of participants, which we find might pose a problem in the future if it is not dealt with.

All other questions yielded prevailingly positive answers, ranging from 67-75% of positive (satisfied or very satisfied) answers, with only exception of "Team satisfaction" where the positive is 42%, but additional 50% are neutral.

When analyzing results at a company level we generated data visualized in Figure 21.

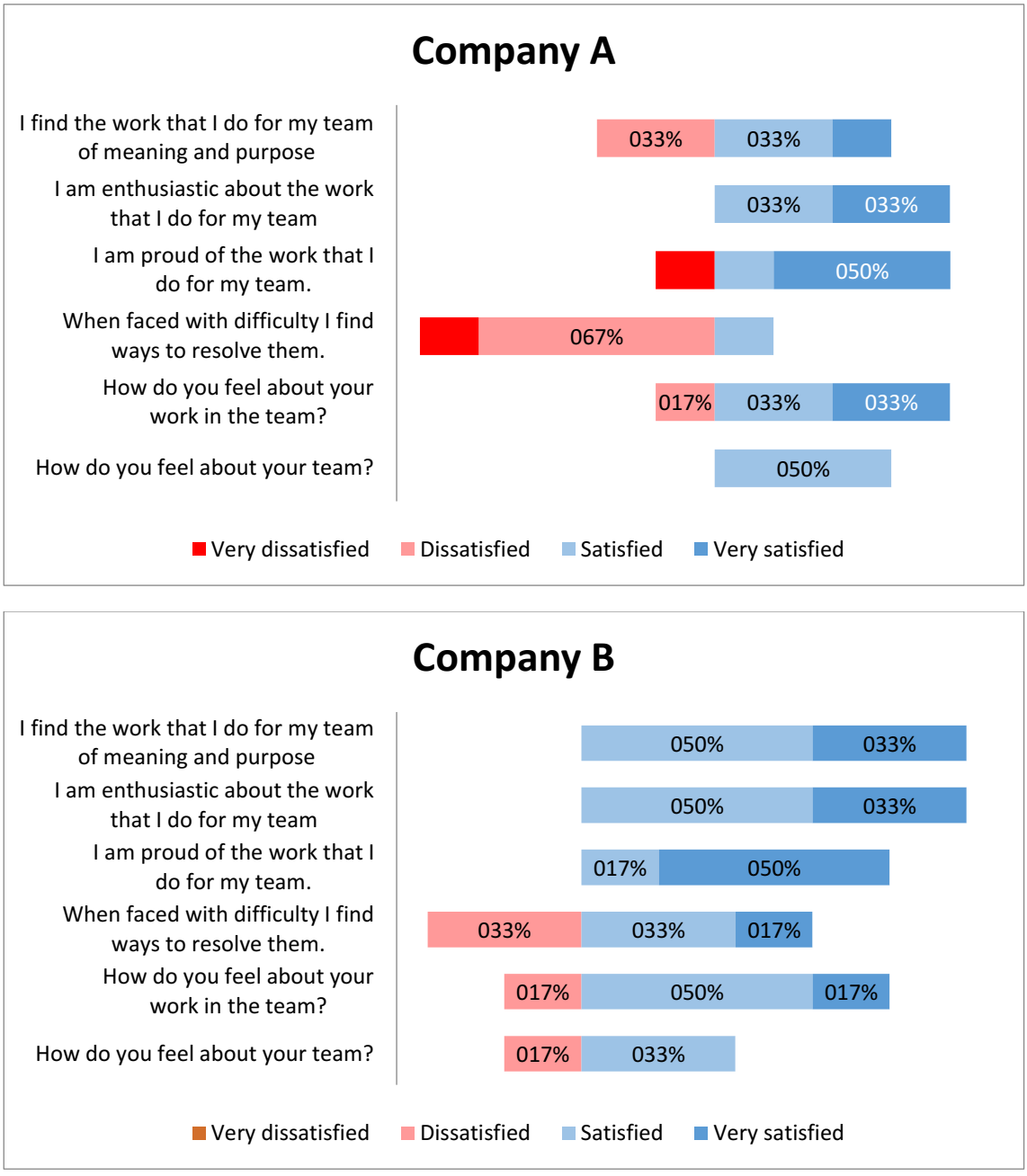

Figure 22: Perceived satisfaction in Agile projects

When comparing the data between the companies, not only does the perceived satisfaction in Company A seem lower, but the extremes are also more pronounced. In order to compare the

companies in absolute terms we created „satisfaction index“ by associating points to answers following the principle:

- Strongly Disagree (1)
- Disagree (2)
- Neither agree nor disagree (3)
- Agree (4)
- Strongly agree (5)

and then dividing it by the maximum possible number of points for the relevant group of participants.

The resulting „satifaction indicators“ came rather close: 67% for Company B, and 69% for Company A. While Company A has more responses which are either „Disatissfied“ or „Very dissatisfied“ it also has more of the „Very Satisfied“ responses actually bringing its index ahead of Company B.

4.5 Conclusions

This research work can be summarized along the following four research questions which were derived from the literature research.

1) *How to describe waterfall and agile software development in practice?*

Our study shows that Company A, the example of agile development method in practice, had a clear and defined production process and stakeholders, but somehow less defined team roles. Even though the process was considered as stable, a further development of the product was discontinued due to insufficient project funding or undefined strategy, and the first release of the product was eventually rarely used.

Compared to this, Company B, which we described as switching from agile to waterfall development, can be described as continuously changing. Its methodologies, structures, financial and other agreements were continually adapted, but the environment stayed focused on the necessity of the product development, and its quality and functionality. In this case the project seemed endangered with its internal problems, and not the external issues.

As a result, a proper application of a methodology (waterfall or agile) as it is described in literature, or a stable environment are not guarantees of a successful project, but rather additional complexity factors in the anyways sensitive processes of software development.

2) *Which factors affect the satisfaction of team members?*

To answer the team-related question, our study shows that the team members in Company A were slightly older and their company affiliation was longer (5 years) than in Company B (1-2 years). Interestingly, both methodologies show that team members are (strongly) satisfied with their jobs and roles. However, in case of Company A or the agile methodology, the requirements to the roles and team responsibilities were perceived as clear and well accepted – even though the first question shows the opposite.

In terms of contracting or employability, there was a notable difference between internal employees and external – to some extent temporary contracted – team members and consultants in both companies. This issue is rather notable on the organizational or administrative side. For example, external and internal employees usually use different systems to track their work efforts, holidays, etc. Yet, this indifference is rather noted from a human resources or people development perspective. In terms of collaborative team performance, this difference was not perceived neither during the projects, nor during the conducted surveys. In reference to the questions about the team affiliation, both internal and external employees react with the feeling of equal team membership.

3) Which factors affect the quality of development?

If we consider that specific training is influencing the quality of development, we may notice the difference in training and specific education of both teams. While in Company A (agile development) team members strongly agreed that they receive the training they need to do their job well, it seems that Company B (waterfall) could invest into raising the trainings for team members. Yet, agile development teams seem to feel secure (and therefore satisfied) in their position if the overall strategy is known and executed.

As a result, we can argue that the choice of methodology has less impact on the quality of product development when compared to the organizational factors, such as budgeting, strategy, communication and other business factors. One of the results of this research work is that the implementation of agile development methodology requires a change not only in managing the teams, but also in providing the right collaboration and communication tools, as well as a mind-shift in managerial expectations. For example, agile development includes a constant delivery of software, which is also regularly presented to the managers or stakeholders; however, the presented software at the review or demo meetings in an agile setting are mostly not final products, but rather a work-in-progress. Hence, managers or stakeholders need to be educated that the process is rather ongoing than sequential.

4) How mature is the development process disregarding the used methodology?

During the literature research we experienced that there are countless proposals to agile development in practice. Later, within the empirical study we noted that proposals emerge within companies and each company seems to have an individual approach to product development. Both company examples show that agile development is a young methodology.

It seems as if there is no perfect or standardized agile process to be implemented in practice. While waterfall seems to exhibit a higher level of maturity in theory, our study results describe agile development as the more mature development method.

5 SUMMARY OF THE THESIS

This chapter includes a summary of the thesis as well as the theoretical and practical implications, limitations, and suggestions for further research

Software development is one of the most propulsive industries generating more and more jobs and businesses on the one hand, and research topics on the other hand. It is not new that software development is among the core operations within companies. Whether it is maintaining the existing software and systems, to so call „run the business“, or introducing new tool and digitalizing the business through novel technologies, to innovate – software development metodologies play a major role in both cases.

We chose the banking industry for our empirical study, since it is considered as an established traditional industry which is currently going through a rational change. Therefore, we assume that software development is core part of the change process as well. In order to receive qualitative answers to a rather unchartered research field, this research work is based on qualitative research methods, including an empirical questionnaire and expert interviews in a comparative case study.

During the initial literature research, we derived the following key insights:

1. Waterfall or the traditional, structured development processes are stronger in definitions of requirements, infrastructure, interfaces, and most important – customer expectations and features in an early stage of development.
2. In a later stage, the costs and efforts to change requirements in Waterfall are exploding. Also, testing and customer feedback is scheduled only at a very late process stage.
3. In contrast to Waterfall, during an initial setup of an Agile process, some notable efforts and time is needed to take the delivery process to an efficient, well-practiced process with different roles and responsibilities.
4. Once the Agile process is set up, the delivery of features and feedback are faster, the customer can be involved into decisions and the complexity of the production seems to be reduced by the sliced product delivery.

As seen during the research of current literature in the field of digital banking, banking innovation or agile development – which can be found in *Chapter 2*; considerable research has been conducted with relation to product, project or process issues. Yet, many topics are left unanswered, such as: which is the beneficial software development methodology in the field of banking, or how does the selection of a methodology affect the satisfaction of the team members, the quality of development, or the process maturity.

In order to answer the stated questions, we have taken a deep insight into two practical cases of software development in banking in *Chapter 3*. Company A is the example of agile development, while Company B deals with a shift from agile to Waterfall. On the one hand,

we analyzed agile methodology which was well set up within a rather unhospitable environment. On the other hand, we discussed a shift from agile towards waterfall in a significantly less structured and defined surrounding. Eventually, we compared the two cases and tried to derive further research questions in order to compare theory with practical examples.

Based on the previous research steps, we have conducted a survey to interview team members from both Company A and Company B. A deeper analysis can be found in *Chapter 4*.

The main results include that: a) agile development is successfully applicable in banking environments; b) there might be cases when it is appropriate to revert to waterfall and c) team members' satisfaction with the company gets reflected in projects.

Although this research work delivers a very broad set of answers to the research questions, we are aware of the limitations of our work, such as:

- We have selected a qualitative research method, which means that we provide descriptive conclusions to our questions, not quantitative data.
- Even though the anonymity of the survey applicants was guaranteed, they were reluctant to criticize the company. This could be a reason for positive answers related to team satisfaction.
- This research work focused on the development process itself, not on the outcomes.

Due to further limitations in time and data, we are aware that our results are not universal. Since some of the examined data is considered as internal data from corporations, the thesis does not provide a fully proven answer to the research questions, but rather an insight into the corporate examples in the CEE region. For example, it was not possible to answer the open question of which method is better for stakeholder management or customer experience.

For further research, it may be interesting to compare the two examples in relation to: process efficiency, number of bugs, team size, costs of development, investment, strategy, and above all the customer perspective.

Essentially, no matter whether software development and related methodologies in the context of banking are analyzed in an academic, commercial or social context, the topics merit further research.

6 REFERENCES

[1] **P. Abrahamsson, K. Conboy and X. Wang**, "'Lots done, more to do': the current state of agile systems development research", European Journal of Information System, no. 18, pp. 281-284, 2009.

[2] **P. Abrahamsson, M. Marchesi, F. Maurer** (Eds.), Agile Processes in Software Engineering and Extreme Programming, Proceedings of the 10th International Conference on Extreme Programming, Sardinia, Italy, 2009.

[3] **P. Abrahamsson, M. Marchesi, G. Succi** (Eds.), Extreme Programming and Agile Procresses in Software Engineering, Proceedings of the 7th International Conference on Extreme Programming, Oulu, Finnland, 2006.

[4] **M. O. Ahmad, J. Markkula and M. Oivo**, "Kanban in software development: A systematic literature review," 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, Santander, 2013, pp. 9-16. doi: 10.1109/SEAA.2013.28

[5] **Ambler S** (2002). Agile modeling – effective practices for extreme programming and the unified process. Wiley, New York

[6] **Anderson, David J.**, Kanban: successful evolutionary change for your technology business. Blue Hole Press, 2010.

[7] **T. Arthur:** "Agile Adoption: Measuring its Worth", 2013., SAS Institute, Inc.

[8] **R. Babineaux and J. Krumboltz**, Fail Fast, Fail Often: How Losing Can Help You Win, New York: Jeremy, P. Tarcher / Penguin, 2013.

[9] **B. Barton**, "All-Out Organizational Scrum as an Innovation Value Chain," System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on, Big Island, HI, 2009, pp. 1-6. doi: 10.1109/HICSS.2009.57

[10] **Beck, K**. Extreme Programming Explained - Embrace Change. Reading, MA: Addison Wesley Longman, Inc. 2000.

[11] **K. Beck et al**. Manifesto for Agile Software Development. http://www.agilemanifesto.org, Apr. 2001.

[12] **A. Begel and N. Nagappan**, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, 2007, pp. 255-264. doi: 10.1109/ESEM.2007.12

[13] **B. Boehm, W. Brown and R. Turner**, "Spiral development of software-intensive systems of systems," Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005., Saint Louis, MO, USA, 2005, pp. 706-707. doi: 10.1109/ICSE.2005.1553673

[14] **Cadle, J., Yeates, D**., 2008. Project Management for Information Systems. 5th ed. Harlow, England: Prentice Hall.

[15] **M. Ceschi, A. Sillitti, G. Succi and S. De Panfilis**, "Project management in plan-based and agile companies," in IEEE Software, vol. 22, no. 3, pp. 21-27, May-June 2005. doi: 10.1109/MS.2005.75

[16] **S. Chopra**, "Implementing Agile in old technology projects," Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014 3rd International Conference on, Noida, 2014, pp. 1-4. doi: 10.1109/ICRITO.2014.7014757

[17] **Ashraf Ferdouse Chowdhury and Mohammad Nazmul Huda**, "Comparison between Adaptive Software Development and Feature Driven Development," Computer Science and Network Technology (ICCSNT), 2011 International Conference on, Harbin, 2011, pp. 363-367. doi: 10.1109/ICCSNT.2011.6181977

[18] **I. Christou, S. Ponis and E. Palaiologou**, "Using the Agile Unified Process in Banking," IEEE Software, vol. 27, no. 3, pp. 72-79, 2010.

[19] **Coad,P.,Lefebvre,E.,DeLuca,J**. (1999). Java Modeling in Color with UML: Enterprise Components and Process. Prentice-Hall.

[20] **D. Cohen, M. Lindvall, and P. Costa**. An Introduction to Agile Methods. Advances in Computers, pages 1–66, 2004.

[21] **O. C. de Melo, S. D. Cruzes, F. Kon and R. Conradi**, "Interpretative case studies on agile team productivity and management," Information and Software Technology, vol. 55, no. 2, pp. 412-427, 2013.

[22] **T. Dingsøyr et al**. (eds.), Agile Software Development, DOI 10.1007/978-3-642-12575-1_1, ©Springer Verlag Berlin - Heidelberg 2010

[23] **T. Dingsøyr, S. Nerur, B. VenuGopal and M. Nils Bred**e, "A decade of agile methodologies: Towards explaining agile software development," The Journal of Systems and Software, vol. 85, no. 6, pp. 1213-1221, 2012.

[24] **T. Dybå and T. Dingsøyr**, "What Do We Know about Agile Software Development?," IEEE Software, vol. 26, no. 5, pp. 6-9, 2009.

[25] **A. Elshamy, A. Elssamadisy**, “Applying agile to large projects: new agile software development practices for large projects”, Proceedings of the 8th international conference on Agile processes in software engineering and extreme programming, June 18-22, 2007, Como, Italy.

[26] **H. C. Estler, M. Nordio, C. A. Furia, B. Meyer and J. Schneider**, "Agile vs. Structured Distributed Software Development: A Case Study," 2012 IEEE Seventh International Conference on Global Software Engineering, Porto Alegre, 2012, pp. 11-20. doi: 10.1109/ICGSE.2012.22

[27] **Taghi Javdani Gandomani, Hazura Zulzalil, Abdul Azim Abdul Ghani, Abu Bakar Md. Sultan and Mina Ziaei Nafchi**, "OBSTACLES IN MOVING TO AGILE SOFTWARE DEVELOPMENT METHODS; AT A GLANCE" Journal of Computer Science, Volume 9, Issue 5, Pages 620-625. DOI : 10.3844/jcssp.2013.620.625

[28] **L. Gren, R. Torkar and R. Feldt**, "Work Motivational Challenges Regarding the Interface between Agile Teams and a Non-Agile Surrounding Organization: A Case Study," Agile Conference (AGILE), 2014, Kissimmee, FL, 2014, pp. 11-15. doi: 10.1109/AGILE.2014.13

[29] **Hassan Hajjdiab and Al Shaima Taleb**, " Adopting Agile Software Development: Issues and Challenges. International Journal of Managing Value and Supply Chains (IJMVSC) Vol. 2, No. 3, September 2011. DOI: 10.5121/ijmvsc.2011.2301 1

[30] **H. A. Henke**, "Spiraling up and down the spiral development staircase [software development process]," Professional Communication Conference, 1994. IPCC '94 Proceedings. Scaling New Heights in Technical Communication., International, Banff, Alta., 1994, pp. 401-405. doi: 10.1109/IPCC.1994.347490

[31] **Kabira, M., Rusub, L**.: A Framework for IT Project Development in a Large Company (2013), Procedia Technology, Volume 9, pp. 687-696

[32] **Kotonya, Gerald, and Ian Sommerville**. Requirements engineering: processes and techniques. Wiley Publishing, 1998.

[33] **T. J. Lehman and A. Sharma**, "Software Development as a Service: Agile Experiences," 2011 Annual SRII Global Conference, San Jose, CA, 2011, pp. 749-758., doi: 10.1109/SRII.2011.82

[34] **M. Lindvall et al**., "Agile software development in large organizations," in Computer, vol. 37, no. 12, pp. 26-34, Dec. 2004. doi: 10.1109/MC.2004.231

[35] **R. Mahdavi-Hezave and R. Ramsin**, "FDMD: Feature-Driven Methodology Development," Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on, Barcelona, Spain, 2015, pp. 229-237.

[36] **N. B. Moe, T. Dingsøyr and T. Dybå**, "A teamwork model for understanding an agile team: A case study of a Scrum project," Information and Software Technology, vol. 52, no. 5, pp. 480-491, 2010.

[37] **N. Oza, F. Fagerholm and J. Münch**, "How does Kanban impact communication and collaboration in software engineering teams?," Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on, San Francisco, CA, 2013, pp. 125-128. doi: 10.1109/CHASE.2013.6614747

[38] **M. Penttinen and T. Mikkonen**, "Subcontracting for Scrum Teams: Experiences and Guidelines from a Large Development Organization," 2012 IEEE Seventh International

Conference on Global Software Engineering, Porto Alegre, 2012, pp. 195-199. , doi: 10.1109/ICGSE.2012.16

[39] **M. Poppendieck**. (2016, July 12), An Introduction to Lean Software Development [Online]. Available: http://www.leanessays.com/2004/06/introduction-to-lean-software.html

[40] **M. Poppendieck, T. Poppendieck**, Lean Software Development - An Agile Toolkit, Addison Wesley, 2003.

[41] **Richman, L**., 2002. Project Management Step-by-Step. New York: AMACOM.

[42] **Royce, Winston W**. "Managing the development of large software systems."proceedings of IEEE WESCON. Vol. 26. No. 8. 1970.

[43] **Schwaber K, Beedle M** (2001) Agile software development with scrum. Prentice Hall, Englewood Cliffs, NJ

[44] **Sommerville, I.**, 2011. Software engineering. 9th ed., Boston, Massachusetts: Pearson Education.

[45] **Stober, T., Hansmann**, U., 2010. Agile Software Development: Best Practices for Large Software Development Projects. Berlin, Heidelberg: Springer.

[46] **K. Sureshchandra and J. Shrinivasavadhani**, "Moving from Waterfall to Agile," Agile, 2008. AGILE '08. Conference, Toronto, ON, 2008, pp. 97-101. doi: 10.1109/Agile.2008.49

[47] **J. Sutherland**, "Future of scrum: parallel pipelining of sprints in complex projects," Agile Development Conference (ADC'05), 2005, pp. 90-99., doi: 10.1109/ADC.2005.28

[48] **Z. Šochová and E. Kunce**, "A Story About Dinosaur Called Mainframe and a Small Fly Agile," in 2012 Agile Conference (AGILE 2012), Dallas, TX, 2012.

[49] **H. Takeuchi and I. Nonaka**, "The New New Product Development Game," Harvard Business Review, 1986.

[50] **K. Terlecka**, "Combining Kanban and Scrum -- Lessons from a Team of Sysadmins," Agile Conference (AGILE), 2012, Dallas, TX, 2012, pp. 99-102. doi: 10.1109/Agile.2012.20

[51] **Tor, M. and Sarfraz, S**. (2013): „SNL: Data Dispatch: Largest 100 banks in the world “. SNL Interactive, S&P Global Market Intelligence. Visited on July 19, 2016 on https://www.snl.com/InteractiveX/Article.aspx?cdid=A-26316576-11566.

[52] **R. Tufail and A. A. Malik**, "A Case Study Analyzing the Impact of Software Process Adoption on Software Quality," Frontiers of Information Technology (FIT), 2012 10th International Conference on, Islamabad, 2012, pp. 254-256., doi: 10.1109/FIT.2012.52

[53] **Wikipedia contributors**. (2016, August 15). Scrum (software development) [Online]. Available:

https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=734632344

[54] **Wood, Stephen, George Michaelides, and Chris Thomson**. "Successful extreme programming: Fidelity to the methodology or good teamworking?." Information and Software Technology 55.4 (2013): 660-672.

[55] **R. Yin**, Case Study Research: Design and Methods, SAGE Publications, 2013.

7 APPENDIX

7.1 Questionnaire

This is an integral overview of the types and interdependencies of the questions asked. Each question was formed so it can be used in an online, highly targeted survey and offline as either a classical survey, or as a part of structured interviews.

Opening Comments

Invitation for the survey was sent to individuals for whom the professional affiliation and experience was established in advance, therefore the invitation was customized and had additional directions included. One example of aforementioned targeting instructions reads as follows:

- "In your case the survey is intended to be focused around your experiences with Company A/B, regardless in which company you were formally employed when working for Company A/B. If you no longer work for the Company A/B, I would ask to try and apply your experiences relating to the time when you worked for Company A/B",

directly referring to the company by name instead of the placeholder.

The text we used additionally to the invitation follows.

- Welcome to the software development practices survey!

  Thank you for agreeing to take part in this important survey, measuring your personal opinion and experiences with different software development methodologies. Questions of this survey will adapt to your personal professional background, but on average it should take 5-15 minutes to complete.

  The results of this survey can be used in an aggregated and anonymous form in a master thesis at University of Ljubljana, Slovenia. In case you want a copy or some more information please contact me at [email]

  Be assured that all answers you provide will be kept in the strictest confidentiality.

Demographic Information

In highly targeted surveys like these it can be deemed unnecessary to list the full scope of demographic information (e.g. there will be no participants 17 and younger, and no participants without at least some college), but we decided to keep the scope in the area of general public, to avoid that any participants feel they belong in an outlying group (e.g. among the oldest, or among the least educated).

- What is your age:
  - 17 or younger

  - 18 to 24
  - 25 to 34
  - 35 to 44
  - 45 to 54
  - 55 to 64
  - 65 to 74
  - 75 or older

- What is your highest level of education?
  - Less than high school
  - Graduated high school
  - Trade/technical school
  - Some college, no degree
  - Associate degree
  - Bachelor's degree
  - Master's degree
  - Other advanced degree (PhD, MD, etc.)

- How long have you been working for the company named in your invitation (whether directly or as a consultant/external on a project)?
  - Less than 3 months
  - 3-6 months
  - 6 months to a year
  - 1-2 years
  - 2-5 years
  - more than 5 years

- With which of the following software development methodologies have you worked in the current company?
  - Waterfall development
  - Spiral development
  - Agile development
  - Other - Write in:
  - Exclusive / None of the above

Employee satisfaction

This set of questions was designed to measure general employee satisfaction with their employer. Each question was mapped to standard five-point Likert scale:

- Strongly Disagree (1)
- Disagree (2)
- Neither agree nor disagree (3)
- Agree (4)
- Strongly agree (5)

The questions in this section are as follows.

- I feel valued as an employee / contractor.
- My job requirements are clear.
- I receive the training I need to do my job well.
- My supervisor cares about me as a person.
- I am satisfied with my current work schedule.
- There is a clear path for job advancement.
- My thoughts and opinions are heard.
- I have job security.
- The Company clearly communicates its goals and strategies to me.

Additionally, in this section there was an option to include comments in answer to the following optional question:

- What suggestions do you have for the improvement of your current company?

Waterfall development experiences

This section appeared in the online version of the questionnaire only if the employee answered that s/he has experience with Waterfall methodologies in the Demographics section of the questionnaire. In the offline version the employee was directed to skip the section if no previous experience with methodologies belonging to this group existed.

This section had the goal of measuring the maturity of the waterfall methodology used, and employee's understanding of elementary building blocks of such methodology.

- Which role(s) have you been performing within the Waterfall methodology in the named company?
    - Developer
    - Business analyst
    - Technical analyst
    - Project manager
    - PMO
    - Release manager
    - Architect
    - Quality Assurance
    - Team lead
    - Solution manager

- Other - Write in:

- In waterfall, generally speaking, project management plan has been prepared, together with other planning documents
  - Yes, its content has been communicated to all the stakeholders, and is available on a collaboration platform
  - Yes, but its content (deadlines, resources, scope) sometimes changes
  - Yes, but its content (deadlines, resources, scope) changes often
  - No, there is a fixed budget, time or scope plan
  - Not relevant for me/ I don't know

- While doing waterfall I think user needs have been well analyzed and understood
  - Yes, the needs are known in advance of implementation
  - Yes, but user needs change during implementation sometimes
  - Yes, but user needs change during implementation often and/or significantly
  - No, I think the user needs are not understood or analyzed enough
  - Not relevant for me / I don't know

- While we work in waterfall, the client is aware of their needs and confirms them by a formal process (sign-off)? Sign-off is done before the development?
  - Yes, sign off is the official start of development
  - Yes, but often development starts before the sign-off
  - In reality development starts well before the sign-off of requirements

- Is there a detailed Functional Requirements Document reflecting user needs in your waterfall implementation?
  - Yes, the document is comprehensive, up-to-date, and free from unresolved, high-risk implications
  - Yes, but the document is either not comprehensive, or up-to-date, or has some high-risk implications
  - Yes, but the document has multiple weak areas
  - No, the document either not exists or is not useful
  - Not relevant for me / I don't know

- How often does the scope change after official sign-off of business requirements in your waterfall implementation?
  - Always

  - Often
  - Sometimes
  - Rarely
  - Never

- Is Systems Design Document focusing on how to deliver the required functionality created in your waterfall implementation?
  - Yes, right architecture for implementing the requirements is well understood
  - Yes, but there are some uncertainties about the right architecture, it is slightly not understood
  - Yes, but there are big uncertainties, or changes to the architecture
  - No, the architecture is not present or not understood at all
  - Not relevant for me / I don't know

- Do Quality assurance staff and end users in your waterfall implementation test the delivered system to demonstrate that it conforms to requirements set out in the Functional Requirements Document
  - Yes, QA are well equipped with test cases, fast to find major bugs and provide a detailed report on the issues
  - Yes, but the test cases are not thorough, or some major bugs get missed, or the testing report is not available
  - Yes, but the QA process has significant problems which causes major bugs to get unnoticed or not reported
  - No, QA does not exist or is not useful
  - Not relevant for me / I don't know

- Are releases in your waterfall implementation published into production environment according to a set procedure, after resolution of problems identified in the Integration and Test phases
  - Yes, releases are well defined, process is known, and release to production seldom causes extended downtime, or need a rollback
  - Yes, releases are defined, but the process is not known, or extended downtime is needed, or rollback is sometimes needed
  - Yes, releases are sometimes defined, but not stable, or not known
  - No, releases are not defined, or are informal
  - Not relevant for me / I don't know

- Are post-production problems in your waterfall implementation documented and resolved according to procedure

  - Yes, all the defects that are noticed are recorded and there is a clear process of assigning them to the appropriate person/team and their fixing
  - Yes, most defects are recorded, and there is a general process for assigning and solving them
  - Yes, defects are recorded, but often are passed around between assignees and usually take a long time to solve
  - No, defects are informally discussed, and sometimes solved
  - Not relevant for me / I don't know

Series of multiple-choice questions was then followed by an opportunity to insert an optional comment on the Waterfall methodology within the company in question

- My general comment about Waterfall development methodology in my current company

Company satisfaction while using Waterfall methodology

This section was shown to the responders who indicated that they had experiences with Waterfall methodologies in the targeted companies.

The section was designed to measure the individual perception of satisfaction with the company and workplace during the time while practicing waterfall methodologies. Responders were asked to evaluate their feelings in this context.

- How do you feel about the company while practicing Waterfall?
  - Very Dissatisfied
  - Dissatisfied
  - Neutral
  - Satisfied
  - Very Satisfied

- How do you feel about your work at the company, while practicing Waterfall?
  - Very Dissatisfied
  - Dissatisfied
  - Neutral
  - Satisfied
  - Very Satisfied

- When faced with difficulty, while practicing Waterfall, I find ways to resolve them.
  - With much difficulty

  - With some difficulty
  - Neutral
  - Easy
  - Very easy

- I am proud of the work that I do for my company while using waterfall.
  - Not proud at all
  - Mostly not proud
  - Neutral
  - Somewhat proud
  - Very proud

- I am enthusiastic about the work that I do in waterfall methodology for the named company
  - Not enthusiastic at all
  - Mostly not enthusiastic
  - Neutral
  - Somewhat enthusiastic
  - Very enthusiastic

- I find the work that I do for the named company, in waterfall methodology, of meaning and purpose
  - Void of all meaning and purpose
  - Mostly without meaning and purpose
  - Neutral
  - Mostly meaningful and purposeful
  - Very meaningful and purposeful

Agile development – general section

This section appeared in the online version of the questionnaire only if the employee answered that s/he has experience with agile development methodologies in the Demographics section of the questionnaire. In the offline version the employee was directed to skip the section if no previous experience with methodologies belonging to this group existed.

This section had the goal of measuring the maturity of the agile methodology used, and employee's understanding of elementary building blocks of such methodology.

- Which Agile methodologies did you use in company named in your invitation?

- Scrum
- XP
- Feature Driven Development
- Kanban
- Other-Write in: _____

Following the selection of methodologies used the responder was presented with a choice of roles within the agile development.

- Which role(s) have you been performing within the agile methodology in the named company?
  - Delivery team member
  - Scrum master
  - Product owner
  - Other – Write in: _____

Were the responder to choose in the previous question “Delivery team member” additional question was shown:

- As a member of agile team in the named company I performed the following activity(s)
  - Development
  - Testing (QA)
  - Analysis
  - Specialist
  - Process Coordinator
  - Kanban Master
  - Team Lead
  - Steward
  - Project manager
  - Other - Write in: _____

As the final part of the general section on agile development the user was presented with an opportunity to express some comments on the agile adoption within the company:

- My general comment about Agile in my current company is: ____

Agile development – requirements management

This section of the questionnaire was created to collect impressions on elements which are generally regarded as indicators of maturity of process with regards to requirements

management. To limit the fatigue of responders the options were limited to “yes” or “no”, depending on the prevailing side. The section was shown if the responder indicated s/he had experiences with agile methodologies.

- Development feature list (backlog) is adequately prioritized
- There is a clear and common understood definition of done/acceptance tests for completed features
- Scope of each release is planned in advance
- Stories (user requirements) are written in a way that describes how a user can benefit from the feature
- Developers and testers work together in story/requirement implementation and acceptance criteria
- Stories/requirements are broken down and small enough to be done in one sprint/iteration
- Epics and/or theme concepts are used to help organize groups of stories.
- Incomplete stories are well-managed (e.g. finished in next iteration)

Agile development – process management

This section of the questionnaire was created to collect impressions on elements which are generally regarded as indicators of maturity of process with regards to process management. To limit the fatigue of responders the options were limited to “yes” or “no”, depending on the prevailing side. The section was shown if the responder indicated s/he had experiences with agile methodologies.

- Each team member creates a record of their tasks to help break down their assignments (e.g. via a tool, owned by the member and updated when needed)
- Regular Daily Scrum Stand-ups occur
- Daily Scrum Meetings are not overly long
- Stories are estimated in points
- Iteration planning meetings occur
- Iterations are loaded to the right capacity
- Iterations do not change length
- Teams come prepared to the iteration planning meetings
- If used, hardening (or stabilization) iterations are scheduled in advance
- End-of-iteration demos occur
- Product is potentially shippable at the end of each iteration
- Blocks (impediments) are resolved quickly

Agile development – quality assurance

This section of the questionnaire was collecting impressions of the responders regarding the quality process and outcome of the agile methodology. The section was shown if the responder indicated s/he had experiences with agile methodologies. The given options were “yes” and “no”, depending on what the respondent felt was prevailing side.

- Testers participate alongside development.
- Dependencies are well-managed
- Defect levels are continuously monitored (low technical debt).
- Unit testing occurs
- Code reviews occur
- Architectural design for the product(s) is understood by the team
- Automated unit and/or acceptance tests are run as part of each automated build
- Team members pair-program at appropriate times
- Everyone that’s needed for this project is assigned, engaged, or available as needed
- Team members work on finishing each iteration as a team, helping each other along the way as needed
- Management sets goals and gives the team freedom to deliver successfully
- Formal written documents are used to supplement rather than replace faster, more informal communication
- Product management or someone in that role is integrally involved
- The team responds to change in a swift, non-bureaucratic way
- Coaching is utilized to help adopt agile practices
- Retrospectives occur periodically
- An appropriate level of action is taken based on retrospective feedback

Company satisfaction while using agile methodology

This section was shown to the responders who indicated that they had experiences with agile methodologies in the targeted companies.

The section was designed to measure the individual perception of satisfaction with the company and workplace during the time while practicing agile methodologies. Responders were asked to evaluate their feelings in this context, and this was intended to be used in contrast with satisfaction while practicing waterfall technologies.

- How do you feel about your team?
    - Very Dissatisfied
    - Dissatisfied
    - Neutral
    - Satisfied
    - Very Satisfied

- How do you feel about your work in the team?
    - Very Dissatisfied
    - Dissatisfied
    - Neutral
    - Satisfied
    - Very Satisfied

- When faced with difficulty I find ways to resolve them.
    - With much difficulty
    - With some difficulty
    - Neutral
    - Easy
    - Very easy

- I am proud of the work that I do for my team.
    - Not proud at all
    - Mostly not proud
    - Neutral
    - Somewhat proud
    - Very proud

- I am enthusiastic about the work that I do for my team
    - Not enthusiastic at all
    - Mostly not enthusiastic
    - Neutral
    - Somewhat enthusiastic
    - Very enthusiastic

- I find the work that I do for my team of meaning and purpose
    - Void of all meaning and purpose
    - Mostly without meaning and purpose
    - Neutral
    - Mostly meaningful and purposeful
    - Very meaningful and purposeful

Your information

In this short section the responders were offered to share their information for subsequent interviews, and thanked for their time and participation in the survey.