Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

Exploring Software Process Variation Arising from Differences in Situational Context

Paul Clarke[1,2], Rory V O'Connor[1,2], David Solan[3], Peter Elger[4], Murat Yilmaz[5], Adam Ennis [1], Mark Gerrity[1], Sean McGrath [1], Ryan Treanor [2]

[1] School of Computing, Dublin City University, Ireland
{paul.m.clarke, rory.oconnor}@dcu.ie, {adam.ennis22, sean.mcgrath43, mark.gerrity2, ryan.treanor2}@mail.dcu.ie,
[2] Lero – the Irish Software Research Centre
[3] FINEOS Corporation
David.Solan@fineos.com
[4] nearForm Limited
elger.peter@gmail.com
[5] Çankaya University, Ankara, Turkey
myilmaz@cankaya.edu.tr

Abstract. The software development process is continuously changing, there is huge pressure to condense release cycles into shorter and shorter timeframes, tools are changing dramatically and companies must continually examine the efficacy of their development process. Attempting to hit a moving target is difficult and it is a decision which can have a major effect in terms of both the end-product and the business. In this paper, we discuss the role of situational context in deciding upon the software development process through the analysis of two case studies. The case studies take a detailed look at the organisational profile and context of each company in turn before we compare and contrast each situational context for factors that may influence the development process. We then compare the processes each company has chosen before our discussion of the role context plays in choosing a 'correct' software development process. While both companies have enjoyed sustained business growth and while both are agile in mind-set, we find that they are in fact quite distinct in their processes, this distinction being driven by their different situational contexts.

Keywords: Agile, SAFe, Situational Context, Software Development Process, Software Engineering

1 Introduction

An evidence-driven, universal set of guidelines on how to approach the software development process is lacking in the literature. This is, in part, due to the complex nature of software development, the many factors of which may constitute a *complex system* [1]. A software development process can be defined as "a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products" [2]. The factors that influence the process can be thought of as the situational context, these factors include the nature of the end-product, volatility of requirements, personnel skill levels, organisational culture and many others. Various software development processes have come to the fore over the years including the Waterfall [3], CMMI [4], Agile methods [5], each with benefits and drawbacks depending on the situational context to which they are applied.

In deciding which process to use and how to apply it, organisations must compare differing approaches without a full appreciation of how that approach fits their particular context. Worse still, companies may arbitrarily choose one, which can lead to a sub-optimal process that may be damaging to organisational performance and software quality [6]. We apply work done previously on producing an initial reference framework [7] and use this framework to analyse the many elements of the software development process in two distinct case studies. The first is on FINEOS - an enterprise software development company developing primarily for the insurance industry [8]. The second is nearForm Ltd. - the world's' largest Node.js consultancy firm [9]. We will look for insight into how situational factors affect the performance of their development process and whether lessons can be learned for other companies managing their software development process.

In structuring the case studies, we adopted the following methodology: Firstly, the companies were invited to present on their software development process and also to indicate how their situational context has informed their software process decisions. This presentation typically lasted around 100 minutes, including extensive opportunity for questions and answers. Secondly, further research was conducted on the participating companies, for example by evaluating their websites and press releases, to build up a well-rounded view of the companies' operating context and reported development process initiatives. Thirdly, the relationship between their respective development process and situational context was examined. In the fourth step, the total learning from both companies was documented and shared with both organisations offered an opportunity for feedback and clarification. The results of this fourth step are consolidated in this paper.

Among the primary findings we conclude that they software development process in the participating companies is a mishmash of the various generic software development approaches. We also find that situational context played a vital role in informing software development process selection and evolution in the participating companies. It is tempting to reach the conclusion that this is representative of the broader software development sphere but we have not collected the broader industrial data that would be required to enable an evaluation of that perspective.

This paper is structured as follows: Section 2 describes in some detail the situational context of the software development process in one company. Section 3 provides equivalent detail but for the second case study organisation. Section 4 will compare and contrast the two situational contexts with a view of finding similarities and points of divergence. Section 5 will compare both software development processes again looking for pertinent similarities and differences. Section 6 contains a discussion and conclusion.

2 Case Study 1 - FINEOS

The first company we will consider is FINEOS; a software development company that delivers enterprise solutions for the insurance industry. The company was founded in 1993 by Michael Kelly, the current CEO, and has since grown to 500 employees and approximately 26,000 software users across multiple countries. Their HQ is in Dublin, Ireland, and they have offices in the US, Poland and Australia. Although they are based in Dublin, they estimate that as much as 50% of their customers are based in the United States. All accident claims in New Zealand, four of the top five insurers in Australia and eight of the top twenty health insurers in America use FINEOS software when processing claims.

The primary specialism of FINEOS is in insurance claims and payment management. Recently, they have moved into offering a full administration suite of components that will cover all aspects of insurance; claims, policy and billing. From this, their software manages six million claim cases annually, involving up to $7 billion.

In the early days of FINEOS, they evolved a plan-driven culture, which was highly organised and structured (and consistent with best practice at that time). Changing customer demands have led FINEOS to implement an Agile culture and environment for their employees and to service their customers. They have incorporated Agile principles (people, process and engineering) for many years, with their first purposeful implementation of agile begin applied to a UX refresh of their system (and implemented in Scrum). FINEOS continues to incorporate the Scrum methodology as one of their core process initiatives. This involves setting up project teams, usually consisting of 7-9 people. They work in iterative sprints, developing elements of the overall system within each sprint cycle. These cycles generally consist of five two-week sprints, with a system release to the customer at the end of the ten weeks. They also include a parallel planning process as part of entering each cycle in line with the need to change requirements. Other elements of the methodology that FINEOS has assimilated into their process are; daily stand-ups, sprint burndowns, Product Owners and Scrum Masters [10]. FINEOS implement Inspect and Adapt sessions regularly to learn from their experiences and implement changes which improve the overall flow of work and deliverables.

In recent years, they have been using the Scaled Agile Framework (SAFe). The SAFe uses a combination of various Lean and Agile concepts, and deals with three levels of an organisation; team, program and portfolio [11]. FINEOS do not implement all elements of the SAFe framework, but only those they deem beneficial for their business. They tend to merge elements of the framework together, for example, integrating multiple roles and assigning them to one person. Their release cycle has shrunk from every four months to at most ten weeks since implementing an Agile approach and as a result a substantial increase of automation has also been seen in the

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

company. They now include a full suite of code checks, with build, test, and delivery automation running several times a day. In terms of personnel skill FINEOS believe in having 'T-shaped' people. This identifies people with an in-depth knowledge of one discipline, and a broad knowledge of other related areas of expertise, which are generally technology orientated in the case of FINEOS [12].

A key factor of their situational context is insurance industry attitudes to data accuracy and security. FINEOS offers the possibility for clients to utilise their cloud-based service. However, this can be a point of contention for customers as traditionally, some in the insurance industry have stored their data in on-site data centres. This is because they may not always trust cloud providers to store their data securely even though there is evidence showing there is adequate security in the cloud [13]. Incidents such as data breaches can greatly damage an insurer's reputation and undermine the public's confidence in the industry [14]. Plus, the integrity of insurance data, including records and calculations, has perhaps inclined insurance software providers to continue in their adoption of traditional methods such as Waterfall [15] which invest heavily in requirements clarification up front. Providing certain conditions are met, such as outlining performance requirements and testing protocols early, Agile methods are proving that they too can offer the type of certainty that traditional methods have proven to delivery [16]. FINEOS focuses on developing a flexible and configurable, modern suite of applications. This contrasts with the fact that some insurance companies that FINEOS engage with are found to be working with legacy systems [17]. Legacy systems are prominent in the insurance industry for a number of reasons, such as the risk associated with replacing the systems, and more commonly the inability to justify the cost of replacing a core system rather than maintaining it [18]. With this in mind, FINEOS has solutions to both replace existing software and to support existing technology [19].

3 Case Study 2 – nearForm Ltd.

The second company we will consider is nearForm Ltd., the World's first Node.js consultancy company. They have grown rapidly from employing 4 people to over 100 people in the last 5 years with revenue growing exponentially. The company was established in 2011 with the vision to change how software is built. Their headquarters is in Tramore, Co. Waterford, Ireland, with staff in the United States, South Africa, France and Romania. nearForm has delivered over 50 large-scale production systems for clients in Ireland, UK and the United States, clients include Intel, The Sunday Business Post and SAP [20]. They offer large enterprises the opportunity to embrace recent disruptive technologies to refactor their monolithic system architecture into a microservices architecture. This allows an Enterprise to change their approach to certain processes and accelerate their development with new tools.

When working on projects with large enterprises, nearForm don't religiously follow certain software development methodologies like Scrum. For example, they don't have the standard role of a Scrum Master, they have an Executive Architect. The difference being - like every member of the workplace - an Executive Architect is directly

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

contributing to code within nearForm, while also adopting roles that may be emulated by a Scrum Master. Rapid and continuous delivery has become a key factor in the modern software development process. nearForm avoid strict Scrum methodology adoption as it might restrict their approach to dealing with clients and, preferring to choose a set of practices depending on the situational context of the client [21].

Given several production releases per day; uncertainty or resource mismanagement must be mitigated as much as possible. They need to be adaptable and be able to apply a modified software development process to any client they work with, they do this by considering factors such as personnel, organisation, business, management, operational, technical, requirements and application. Beginning with Personnel, nearForm boasts an impressive staff turnover of low single digit numbers annually while keeping team sizes across all projects to about 5 or 6. This means there is less need for documented process descriptions or product architecture. The presence of an Executive Architect instead of a conventional project manager allows their workforce to be more adaptable to change, by increasing the level of communication within the team. Cohesion is helped having all employees write code and get experience on all aspects of a project. This gives their teams the ability to work well with undefined elements and objectives they encounter from their clients. Elger believes that the best developers may be up to 30 times more productive than the average developers [21]. Because of this, skill, experience and productivity are aspects which benefit nearForm throughout their development process.

When looking at changeability, it is important that their process is flexible and adaptable as projects are often subject to drastic changes. For this reason, nearForm tend to apply Lean/Agile characteristics to their development process. Product quality is a key focus in the company. As quick delivery is crucial to be successful Time to Market is a factor taken into account when deciding on a delivery process and the tools they use. Deployment infrastructure like Docker has facilitated nearForm in being as efficient as possible when it comes to deployment.

The degree of risk is a big factor, they categorise projects by the strength of risk of data being leaked and in turn, its impact, and while they are not averse to risk, they must be conservative when it comes to delivery for a client, which has led to a strange dichotomy in the firm. For this reason, they tend to use tools and frameworks with a proven track record. Another interesting method used by nearForm is that of breaking conventional sprint constraints; the team will abandon a sprint, fix an urgent issue or scrap a feature bundle so as to adapt to new changes in their understanding or requirements. This may be unusual among generic software development processes found across a lot of firms. nearForm also avoids big bang integration by continually integrating new software. This helps to increase the efficiency of the entire development process.

Clarke, P. M., O’Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O’Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

Typically, nearForm commence a project with a *concept* sprint involving the client and all team members. Then an ideation stage which consists of workshops and scope meetings where developers from nearForm and the Product Owner meet to identify the features required. During this stage nearForm maintain that the responsibility of the project is to be shared to ensure the client's interest in what is going to be deployed and to build knowledge of the product from the client-side. The team work from a technical backlog with regular assessments to ensure the schedule. Open-source tooling allows the organisation to rapidly deliver high quality deployments. This organisation’s process has a strong focus on the implementation of disruptive technologies in using a dynamic language like Node.js and also the Docker container engine that enables continuous deployment of an individual microservice solitarily, without disturbing the architecture as a whole.

4 Compare and Contrast of Situational Contexts

Personnel: As mentioned, FINEOS puts an emphasis on the importance of having T-shaped employees in their company. With employees that are knowledgeable in a given discipline, they are easily able to assign appropriate roles from the Scaled Agile Framework. When organising teams, they implement the Scrum methodology with team size 7-9 members, which are multi-functional, and are mostly co-located with the array of offices they have worldwide. Each of the team members knows exactly what the goal of the team deliverable is and they balance work activities across team members to ensure delivery. nearForm, in contrast, do not use the Scrum approach [21]. This can be partially attributed to nearForm’s commitment to their open-source software (OSS) community engagement model [22]. When employing new staff, nearForm use this OSS model to recruit rather than other conventional methods like a recruitment agency. This approach ensures they only hire staff with an active interest in the wider scene and who code at the level expected of nearForm employees.

Organisation: With 500 employees FINEOS is a medium-sized organisation, it is multi-national with offices spanning worldwide. Different offices are oriented around separate fields. As previously mentioned, the location of their offices has an effect on the teams, as some but not all are co-located. The organisational size of nearForm is much smaller at just 100 employees but this figure could be a reflection of the organisations age. Their HQ is located in Waterford, Ireland but like FINEOS, are international in scope. Low organisational size can lead to efficient communication in Agile methodology methods through frequent virtual stand-up meetings and a reduction in any communication breakdown between hierarchical personnel with a lessened organisational size [23].

Business: The time to market for the applications that FINEOS develops is important. They are mainly influenced by constraining factors; the expectation of their sales force to output a quality product as quickly as possible and the expectation from their clients to have a high quality and secure system when the project is finished. Their current procedure, which consists of a major system release at approximately ten-week

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

intervals, is what they deem most appropriate with the given constraints (though they continue to aggressively reduce release timeframes). Timescales are even more constrained at nearForm. They use Node.js, their deployment infrastructure uses the Docker container platform and their microservices architecture all assist in the implementation of rapid delivery for an even shorter time to market. nearForm operate in weekly sprints where at the end of each sprint, the team consults with the customer to confirm that the project is moving in the correct direction. Frequent deliveries help gain customer satisfaction, brand loyalty and critically, sales revenue.

Management: Over the course of 20 years FINEOS has been able to work with large international customers and boast high customer satisfaction [19]. The global customer presence that FINEOS has built in an industry like insurance will progress their business continuity as new customers will demand reliability. nearForm on the other hand have built their reputation via the Node.js and OSS community, so there is an underlying philosophy among the personnel on emerging technologies. This helps the organisation progress in unison without conflict on strategic decisions. This deep involvement in the Node.js community is a large factor in their continuity as one of the leading organisations in their field.

Operational: FINEOS has implemented an Agile methodology due to customer and internal demand; customers began to request the ability to adapt requirements more frequently and receive more frequent deliveries of software, thus avoiding the extended time delays that are sometimes associated with traditional software process approaches. nearForm have gone further in that they have no formal process but rather a set of practices (team, distributed workforce, Key Performance Indicators [KPIs]) that are varied from project to project. However, they also operate in a fast-moving market where clients' demands can change throughout the project lifecycle. Flexible process design in nearForm allows rapid changes to be made on projects based on the changing project context.

Technical: FINEOS uses a modern technology stack consisting of leading languages such as Java, HTML5 and JQuery. The tools they use in development consist of the most popular tools used in the industry [19]. Many are open source, e.g. Selenium and Tomcat. FINEOS promote technology learning and adoption via innovation days as part of their culture where teams try out new technologies which can often be adopted into their mainstream. nearForm are also using a very modern technology stack. Every year nearForm host NodeConf EU - a key Node.js event in Europe - providing a forum for the Node.js community. Despite the embedded appreciation for this dynamic technology, there is no dogmatic reason for its use, alternative technologies may accomplish the client's needs depending on context.

Requirements: Since the implementation of Agile methodology, FINEOS has supported the ability to change requirements throughout the development process. This is a factor in why they provide a highly customisable solution for their customers. It is also a reason for the focus on shortening their iterations length as it gives clients a clear picture of the work in progress allowing them to identify any changes they might want to make [24]. nearForm take a slightly different approach; during the ideation stage, client-side Project Owners are incorporated in the

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

workshops that take place to ensure that the client shares the project responsibility for requirements. Clients might seem to demand the 'moon on a stick' and as all the personnel of nearForm are actively involved in code, there is a consistent understanding scoping and cost implications.

Application: The applications developed by both companies are not safety critical applications, i.e. there will not be a loss of life directly attributed to their software. Both companies risk repercussions from reputational damage from data loss. This pressure is arguably higher for FINEOS as the expectations for data integrity and security are very high in the insurance industry, as is product performance, scalability and reliability. There is also pressure on software quality at nearForm, with a need to prove quality to risk averse clients. They use Continuous Delivery systems, drip-fed changes to production and the breakdown of Docker containers to help achieve this high product quality. A further key aspect of *application* relates to the nature of the software being product-based or project-based. In FINEOS, the same product (or variations on the same product) is delivered to most customers, with the necessary demand therefore for product-level consistency and the need for an upgrade path being very high. Building software products which are upgradeable and considers the cost of maintenance and total ownership is necessarily characterised differently to once off project based development. This requires that the process is reliable to the extent that the software product will work in a broad range of settings, across a number of different revisions and patch levels, and for a wide number of diverse users. Whereas in nearForm, much of the development is of a consulting, project-based nature which does not incur the same type of product-level reliability demand.

5 Software Development Process Comparisons & Contrasts

In FINEOS the software development process is heavily oriented around the Scaled Agile Framework. They only incorporate elements of the frameworks that they deem necessary. This process is in the form of five two-week iterations with a system release at the end of the ten weeks... Similar to FINEOS, nearForm don't follow any strict framework, but instead take elements of multiple approaches and adapt them to best suit their client's needs, mostly focusing on Agile and Lean principles. Their process tends to differ more from project to project, as they work across multiple industries whereas FINEOS work strictly with insurance.

Since FINEOS' move to Agile there has been a growing focus on the importance of automation using open source tools such as Selenium, Junit and Jenkins. With iterations becoming shorter, the level of automation has increased. As with FINEOS, nearForm believe in regular and automated testing, while aiming to reduce Big Bang testing. Regression testing is something FINEOS emphasise, as clients will not tolerate receiving software updates that have lost functionality in other areas. Regression testing is also carried out meticulously at nearForm - as they introduce a microservices architecture for a client, they must ensure that all previous functionalities are working.

In FINEOS, cycles generally consist of five two-week sprints with a system release after ten weeks. Iteration lengths are shorter in nearForm, generally being one week in

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

length. The reason for this is it limits the amount the team can go off-track while working on a project. At the end of each week, nearForm consult with the clients and receive feedback. One major difference is - FINEOS typically never break an iteration due to arising issues but nearForm will break sprints, make changes and begin another sprint if something changes in the meantime. FINEOS are perhaps more likely to definitively and explicitly define requirements prior to coding for some aspects which require high degrees of accuracy e.g. payment calculations. The FINEOS requirements may simply be more *definable* up front, not benefiting so much from elevated levels of experimentations through successive rapid sprints.

FINEOS only use the elements of SAFe that they deem relevant, nearForm are even more flexible in terms of following frameworks; they loosely interpret Agile and Lean methodologies. Both companies will often change depending on the client. A key difference between the two companies is that FINEOS tend to assign industry standard roles to their employees that correspond to the roles outlined in the SAFe/SCRUM. They only implement roles they deem necessary, and may appoint multiple roles from the SAFe to a single employee. For instance, FINEOS makes use of Scrum Masters, whereas nearForm don't use Scrum Masters, but instead have Executive Architects.

In terms of teams, at FINEOS they generally consist of about 7-9 people. Team size remains approximately constant in nearForm, where teams consist of about 5 or 6 people. However, there are differences with team structures. In FINEOS, the roles within the teams are well-understood and the team members know what is expected of them. The members of the team often have very different roles and FINEOS is evolving towards T shaped people where individuals play different roles as required within the scrum. For example, there will be a Scrum Master and not every team member will be expected to write production code, some will write test code and aspire to production code. This differs from nearForm, where teams are not as structured, every member of the team contributes to the code written. Members' roles may change across a project as they contribute to different parts of the code. Teams in nearForm are separated by different projects as opposed to different roles like in FINEOS. This observation perhaps highlights the key role of product versus project development as a key situational context constraint. Both companies are affected by the triple constraint of time, budget and quality when it comes to their clients' requirements. Given FINEOS' team structure this constraint may be difficult to manage; for example, the sales team may want new features as quickly as possible, but having limited knowledge of coding, they may not understand the time or resources needed for these features to be created. This is less of a problem in nearForm as everyone is actively contributing to the code – a key innovation in nearForm but perhaps not commonplace across the industry as the skills involved in selling might be considered very different to those required for software development itself.

Both companies differ slightly in regards to the impact of risk on their processes. With FINEOS, they are dealing only with insurance companies, which means it is crucial that any processes are regimented and always held to the same standard. As nearForm work with clients in different industries, they must assess each project individually in

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

terms of risk. The result of this assessment can drastically alter their process from one project to another. To do this, they categorise projects by colour and alter their process where necessary. For instance, if a project is categorised as 'purple' it is deemed to be a high-risk project and depending on the case, the team may completely change their usual process and opt for a Waterfall-like approach - even if it is at the expense of efficiency, time, or cost. FINEOS do not adopt this freedom in their process, perhaps owing to the product development nature of their business. FINEOS may also have more long term customers on their books, giving rise to a more flow of new feature demands.

Technology and tooling also have a profound impact on the processes used by both organisations. In FINEOS, the tools used are open source, modern and generally widely used within the industry meaning staff remain current and skills are available. They take advantage of the open source community but do not let it restrict their ability to work with clients. Heavy automation and top-end software results in quicker iterations and a faster overall process. However, they do not directly mandate contributions to open source as an element of their job description or hiring process. FINEOS use open source tooling and software, but nearForm are even more committed; through a combination of recruitment via open source platforms, encouraging and sponsoring staff to work on personal projects or actively contributing to various communities. As industries vary from client to client with nearForm, when taking on a new project, they need to make sure they have the right tools at their disposal before beginning. Any software or tools they use must be tried and tested beforehand. This is something they feel very strongly about. nearForm have, on occasion, turned down projects when they were not confident enough in the tooling required. This is how heavily tooling is factored into their process; it can be a deciding factor before looking at anything else.

6 Discussion and Conclusion

In this comparison, we have seen in detail the situational context and software development processes in two companies. Both organisations can be considered to be highly successful, each witnessing strong and sustained business growth in recent years. In our investigations, we have learned that there is a pressure on software release cycles in industry – even shorter cycles are becoming the norm, releases can now be as often as multiple times per day [25]; and while multiple daily releases is not always the practice in nearForm, the operating reality is that automation, tooling and practice have moved to the point where such pace of delivery is possible, though clearly this speed of delivery may come at some cost as the basic long-established principle of balancing time, cost and quality constraints remains very much in effect.

To consider the case of FINEOS, a long term successful development firm, a key aspect of their situational context is the need for reliability and high quality in their software product: yes they are proven long term innovators in their field, but their's is a field that would not suffer absences of quality, especially if long-term client data was to become corrupted or if financial transactions were to be miscalculated or under-/over-paid. And while nearForm also produce software of high quality, their reality is that they can often be developing bespoke solutions, under time-and-materials based contract arrangements, and where their clients expect very frequent deliveries of software as a means to explore their own needs. This critical perspective in a situational context may alone largely determine a *minimum deployment frequency*. Where clients have a fundamental operating need for large volumes of personal and financial data over a long term and where multiple clients work with a common evolving product code base, highly reliable software may be more important than highly frequent releases. Though clearly, FINEOS have demonstrated a firm commitment to lowering delivery cycles and radically so. Perhaps therefore, in the longer term future, businesses such as theirs will move ever closer to rapid delivery times..

This micronising of software projects would previously have been impossible without the latest advances in tooling and automation and has led to benefits, such as; improved performance testing and improved quality measurements [26]. Both companies have had great success but without adhering strictly to any single methodology, Agile or otherwise. In fact, a key factor to their success cited by both has been the ability to pick and choose aspects of various generic frameworks based on situational factors such as industry, customers' demands or even on a per-project basis as in the case of nearForm.
Certainly at this point in time the selection of a software development process seems to be more of an art-form than a science, we have shown in this short paper a large number of differing aspects in situational context between two companies with similar objectives. We believe the situational factors are more important than perhaps some companies give them credit for, some companies may focus on the process itself rather than explicitly on the many factors external to the process that can shape outcomes and ultimately the success of the project. With so many situational factors exercising varying degrees of influence on a richly varied generic software development process landscape, we conclude that the selection and evolution of a software development process must be a challenging and constantly evolving concern for software development companies. This finding is consistent with some of our earlier theoretical and empirical contributions [1], [27-28]. Even within individual domains of interest, for example safety critical software development, we have found that a significant degree of situational context and process variation may arise [29].

We further suggest that there is an absence of meaningful assistance for companies at the present time – it is simply not possible for companies, especially those fighting for survival, to allocate large amounts of time to research the myriad of software development approaches that continue to present on the broader landscape. And this task of researching the available process alternatives must be frustrated by the fragmented process terminology, where in previous basic research we have found that individual concepts can be branded using many different terms [30-31]. Yet ironically, an absence of attention to the various available techniques could be damaging to the success of any given firm. For example, those companies who have underappreciated the power of continuous integration might find themselves an in economically untenable position as their competitors raise quality and increase release iterations.

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

There was evidence in the two case studies that there is an appetite, indeed a basic business requirement, to adapt the software development process. Sometimes this is a wholesale process change as in the case of FINEOS moving from a traditional Waterfall approach to Scrum, and later to incorporate SAFe. Other times it is tweaking a process to suit the needs of individual projects, as was seen in the case of nearForm. This observation is evidence that software development firms are continually changing their development processes, some times in small incremental and highly specific ways, other times in larger steps. This finding is consistent with earlier industry-based studies conducted by the authors [32]. The type of process evolution we have found can be termed *process reflexivity*, and in earlier work we have examined the relationship between this reflexivity and business performance [33], finding a positive association between these two phenomena. Therefore, the findings from the two case studies reported on in this paper are consistent with earlier related findings.

In selecting a software process a company must arbitrarily judge which contextual factors are most important and given the great number of process frameworks and factors at play, trial and error may have been the unfortunate strategy for some companies up until this point. The task of choosing an optimal process is hugely challenging given the vast array of factors and sub-factors [7]. The company must work against a multi-dimensional problem, balancing key aspects like (changing) technology, business pressures, customer expectations and personnel. Adding to the complexity is that perhaps none of these situational factors remain constant, changing and morphing continuously with repercussions for projects and products. It is perhaps then no surprise that these companies have either only selectively followed aspects of SAFe – in the case of FINEOS or have largely constructed their own processes using sporadic elements of Agile – in the case of nearForm.

We would suggest this paper puts forward some strong evidence that there is not likely to be any one-size-fits-all software development process solution for a software company. Furthermore, that the way in which process is chosen and implemented should be done with careful consideration to the most pertinent aspects of the individual situational context, perhaps on a per-project basis. Furthermore, at the present time, it seems that software process enactment is highly individualised to individual settings and that despite the advice of software framework, model and process creators, companies simply chop and change off-the-shelf processes to bring them into closer harmony with their own perceived needs.

Acknowledgement. This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 to Lero – The Irish Software Research Centre.

References

1. Clarke, P., O'Connor, R. V., Leavy, B.: A Complexity Theory viewpoint on the Software Development Process and Situational Context. In: Proceedings of the 2016 International Conference on Software and System Process (ICSSP 2016), IEEE, San Francisco, CA, USA (2016)

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

2. CMMI Product Team,: CMMI for systems engineering/software engineering/integrated product and process development, version 1.02, CMMI-SE/SW/IPPD, v1.02. Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pa. (1993).
3. Royce, W.: Managing the development of large software systems: Concepts and techniques. Presented at the (1970).
4. CMMI Product Team, CMMI for Development, Version 1.2, Software Engineering Institute, CMU/SEI-2006-TR-008. Pittsburgh, PA (2006).
5. Fowler, M., Highsmith, J.: The Agile Manifesto. Software Development (2001).
6. How would I know how badly we are losing out through sub-optimal software development. David Consulting Group (2015).
7. Clarke, P., O'Connor, R.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology. 54, 433-447 (2012).
8. Leader in Life, Accident & Health Insurance Software, https://www.fineos.com/.
9. Node.js Consulting, Training, Co-Development & Micro Services, http://www.nearForm.com/.
10. Blom, M. 2010. Is Scrum and XP suitable for CSE Development? In: International Conference on Computational Science, ICCS 2010, May 31 - June 2, Computational Science, University of Amsterdam, The Netherlands.
11. Vaidya, A. 2014. Does DAD Know Best, Is it Better to do LeSS or Just be SAFe? Adapting Scaling Agile Practices into the Enterprise. In: Thirty-Second Annual Pacific Northwest Software Quality Conference 2014, October 20-22, World Trade Center Portland, Portland, Oregon.
12. Oskam, I. 2009. T-shaped engineers for interdisciplinary innovation: an attractive perspective for young people as well as a must for innovative organisations. In: SEFI 37th Annual Conference, July 01-04, Rotterdam, Netherlands.
13. Zhang, X., Du, H., Chen, J., Lin, Y., Zeng, L. 2011. Ensure Data Security in Cloud Storage. In: 2011 International Conference on Network Computing and Information Security, May 14 - 15, Guilin Park Hotel, Guilin, China
14. Financial Crime Task Force. 2016. Issues Paper on Cyber Risk to the Insurance Sector. International Association of Insurance Supervisors (IAIS).
15. Owen Williams October 21, 2013. It's Time for Insurers to Shift from Waterfall to Agile [Online]. Available from: http://iireporter.com/its-time-for-insurers-to-shift-from-waterfall-to-agile/
16. Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L., and Zelkowitz, M.2002. Empirical Findings in Agile Methods. In: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002, August 4–7, Chicago, IL, USA.

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

17. Kumar, R., Fareign, G.D,. Cullen, M,. Cadavez, J,. and Prasad, K. 2011. Maximizing the Business Value from Silos: Service based Transformation with Service Data Models. In: India Conference (INDICON), 2011 Annual IEEE, 16-18 December BITS Pilani, Hyderabad Campus, Hyderabad, India
18. Henry, K. 2013. The Modernization Problem, Part 1: Government & Management of Enterprise IT, ISACA Journal, Volume 1. pp.49-51.
19. Solan, D: FINEOS Software Development Presentation, DCU. 23rd Feb 2017
20. O'Brian, C.: Eran Hammer joins Waterford firm nearForm, http://www.irishtimes.com/business/technology/eran-hammer-joins-waterford-firm-nearForm-1.2244425.
21. Elger, P: nearForm Software Development Presentation, DCU. 6th Mar 2017
22. Hertel, G. Niedner, S. and Herrmann, S.(2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. Research Policy, [online] 32(7), pp 1159-1177.
23. Colombo, M., Piva, E., Rossi-Lamastra, C.: Open innovation and within-industry diversification in small and medium enterprises: The case of open source software firms. Research Policy. 43, 891-902 (2014).
24. Paasivaara, M. and Lassenius, C. (2006). Could Global Software Development Benefit from Agile Methods?. In: IEEE International Conference on Global Software Engineering. Helsinki: IEEE, p.109.
25. Larman, C., Basili, V.: Iterative and incremental developments. a brief history. Computer. 36, 47-56 (2003).
26. Dustin, E., Rashka, J. and Paul, J. (1999). Automated Software Testing: Introduction, Management, and Performance. 1st ed. Addison-Wesley Professional, p.37.
27. Clarke, P., O'Connor, R.V.: Changing situational contexts present a constant challenge to software developers. In: Proceedings of the 22nd European and Asian Conference on Systems, Software and Services Process Improvement (EuroSPI 2015), CCIS (Vol. 543), pp. 100-111, 30 September - 02 October 2015, Ankara, Turkey. (2015).
28. O'Connor, R.V., Elger, P., Clarke, P.: Exploring the impact of situational context: A case study of a software development process for a microservices architecture. In: proceedings of the International Conference on Software and Systems Process (ICSSP), Co-Located with the International Conference on Software Engineering (ICSE), pp. 6-10, DOI:10.1145/2904354.2904368. (2016).
29. Nevalainen, R., Clarke, P., McCaffery, F., O'Connor, R.V., Varkoi, T.: Situational Factors in Safety Critical Software Development. In: Proceedings of the 23rd European and Asian Conference on Systems, Software and Services Process Improvement (EuroSPI 2016), 13-16 September 2016, Graz, Austria. (2016).
30. Clarke, P., Mesquida Calafat, A.L., Ekert, D., Ekstrom, J.J., Gornostaja, T., Jovanovic, M., Johansen, J., Mas, A., Messnarz, R., Najera Villar, B., O'Connor, A., O'Connor, R.V., Reiner, M., Sauberer, G., Schmitz, K.D.,

Clarke, P. M., O'Connor, R. V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., … Treanor, R. (2017). Exploring Software Process Variation Arising from Differences in Situational Context. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz (Eds.), Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings (pp. 29–42). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_3

Yilmaz, M.: An Investigation of Software Development Process Terminology. In: Proceedings of the 16th International Conference on Software Process Improvement and Capability dEtermination, pp. 351-361. (2016)

31. Clarke, P. et al:: Refactoring Software Development Process Terminology through the use of Ontology. In: Proceedings of the 23rd European and Asian Conference on Systems, Software and Services Process Improvement, 13-16 September 2016, Graz, Austria. (2016)

32. Clarke, P., O'Connor, R.V., Yilmaz, M.: A hierarchy of SPI activities for software SMEs: results from ISO/IEC 12207-based SPI assessments, In: proceedings of the 12th International Conference on Software Process Improvement and Capability dEtermination (SPICE 2012). CCIS 290/2012. pp.62-74. (2012).

33. O'Connor, R.V., Clarke, P.: Software Process Reflexivity and Business Performance: Initial results from an empirical study. In: Proceedings of the International Conference on Software and Systems Process 2015 (ICSSP 2015), pp. 142-146, ACM SIG on Software Engineering. Tallinn, Estonia, August 24-26. (2015).