6-18-2022

INTEGRATING MACHINE LEARNING WITH SOFTWARE DEVELOPMENT LIFECYCLES: INSIGHTS FROM EXPERTS

Samuli Laato
University of Turku, sadala@utu.fi

Matti Mäntymäki
University of Turku, matti.mantymaki@utu.fi

Matti Minkkinen
University of Turku, matti.minkkinen@utu.fi

Teemu Birkstedt
University of Turku, teemu.birkstedt@utu.fi

A.K.M. Najmul Islam
LUT University, najmul.islam@utu.fi

See next page for additional authors

Authors

Samuli Laato, Matti Mäntymäki, Matti Minkkinen, Teemu Birkstedt, A.K.M. Najmul Islam, and Denis Dennehy

INTEGRATING MACHINE LEARNING WITH SOFTWARE DEVELOPMENT LIFE CYCLE MODELS: INSIGHTS FROM EXPERTS

Research Paper

Samuli Laato, University of Turku, Turku, Finland, samuli.laato@utu.fi

Matti Mäntymäki, University of Turku, Turku, Finland, matti.mantymaki@utu.fi

Matti Minkkinen, University of Turku, Turku, Finland, matti.minkkinen@utu.fi

Teemu Birkstedt, University of Turku, Turku, Finland, teemu.birkstedt@utu.fi

A.K.M. Najmul Islam, LUT University of Technology, Lappeenranta, Finland, najmul.islam@lut.fi

Denis Dennehy, Swansea University, Swansea, Wales, denis.dennehy@swansea.ac.uk

Abstract

This paper examines the challenges related to integrating machine learning (ML) development with software development lifecycle (SDLC) models. Data-intensive development and use of ML are gaining popularity in information systems development (ISD). To date, there is little empirical research that explores the challenges that ISD practitioners encounter when integrating ML development with SDLC frameworks. In this work we conducted a series of expert interviews where we asked the informants to reflect upon how four different archetypal SDLC models support ML development. Three high level trends in ML systems development emerged from the analysis, namely, (1) redefining the prescribed roles and responsibilities within development work; (2) the SDLC as a frame for creating a shared understanding and commitment by management, customers, and software development teams: and (3) method tailoring. This study advances the body of knowledge on the integration of conceptual SDLC models and ML engineering.

Keywords: Machine learning, AI, information systems development, software development lifecycles, SDLC

1 Introduction

The proliferation of machine learning (ML) in information systems (IS) imposes novel challenges for system development practices (Bawack and Ahmad, 2021; Akkiraju et al., 2020; Ishikawa and Yoshioka, 2019; Laato et al., 2022). Oftentimes contemporary ML development requires input from specialised developer roles, such as data engineers and data scientists (Jüngling et al., 2020). ML systems' reliance on data, as well as other intrinsic phenomena including the inscrutability of the models (Asatiani et al., 2021), the models being probabilistic as opposed to deterministic and negative unintended consequences pose challenges for their development, governance and management (Holstein et al., 2019). In fact, prior literature (Jüngling et al., 2020) has conceptualised ML development to be a distinct area of software development, meaning that existing information system development (ISD) approaches may have to be tweaked to accommodate ML development.

Recent literature has called for more research particularly on how to integrate the work of data scientists to software development lifecycle (SDLC) models (Ishikawa and Yoshioka, 2019; Jüngling

et al., 2020). In brief, SDLC models describe the stages that an IS goes through during its development. SDLC models can be seen to provide conceptual support for the development team, but also assist non-technical personnel (such as customers or upper management) to follow the development. In general, SDLC models give structure to software development and are a necessary support especially for larger and more complex projects (Turetken et al., 2011). Due to the lack of conceptual models for supporting ML development, following, managing, and governing ML development processes may be challenging for project managers and product owners (Ishikawa and Yoshioka, 2019). This can lead to several negative outcomes, one of which is the production of ML models with inherent biases. One of the most famous real world cases where an AI system failed was Microsoft's Tay chatbot, which learned through conversing with Twitter users, and started producing racist tweets within 24 hours (Schlesinger et al., 2018). Since then, several advances have been made in counteracting such occurrences, for example, in the field of AI governance (Mäntymäki et al., 2022; Minkkinen et al., 2022), but literature on the management of ML development practices over the entire life cycle of an IS remains at its infancy (Ishikawa and Yoshioka, 2019; Jüngling et al., 2020). Against this background, we address the following research question (RQ):

What are the challenges related to integrating ML systems with SDLC models from information systems development practitioners' vantage point?

To address this question, interviews were conducted with 19 highly experienced IS professionals. Our results contribute to ISD by advancing understanding of the challenges related to managing ML model development as part of the entire SDLC (Amershi et al., 2019; Jüngling et al., 2020). The remainder of the paper is structured as follows. The related work provides an overview of SDLCs and ML in the context of ISD. This is followed up by the research methodology and data collection and analysis approach. We then present our findings followed up by discussion and conclusions.

2 Related Work

2.1 Software development life cycle models

SDLC models depict the involved development activities and their mutual relationships in the ISD process. SDLCs are mainly concerned with concrete software development, and do not take into related processes such as change and release management (Dennehy and Conboy, 2019; Raghunath et al., 2010). Instead, these are discussed at a higher level of abstraction, mainly the overall project management (ibid). Due to variance and complexity of software projects, pre-designed SDLC models are rarely followed strictly (Giardino et al., 2015). Instead, they provide an overall framework which can be adopted and adapted depending on the specific needs of the project at hand.

Development of SDLC models in IS is rooted in the seminal literature of Royce (1970) and Rubin (1970). Since then, sequential SDLC models have remained in use among upper management while development teams have almost universally moved towards iterative approaches such as scrum (Srivastava et al., 2017). However, no single SDLC model can be considered a 'silver bullet' (Brookes, 1987) due to the highly unpredictable, multifaceted, and context-laden environment of software development (Lyytinen and Rose, 2006). Hence, choosing between different SDLCs is a context-dependent decision (Baseer et al., 2015).

Previous literature has developed categorizations to classify different SDLCs. For example, Rani suggests five categories, namely, (1) waterfall models; (2) V-models; (3) iterative models; (4) spiral models; and (5) agile models (Rani, 2017). A literature review by Baseer et al. (2015), in turn, identified 20 categories of SDLC models, many of which were developed before the year 2000, and are now considered obsolete. Moreover, within the SDLC categories, there is variance in the included steps. For example, some waterfall models include as many as 13 stages, while others only include 6 (Mantei and Teorey, 1989).

Since ISD rarely follows a predictable rigid pattern, the steps in the SDLC often need to be returned to (Paasivaara et al., 2012). One of the first popular models to present an iterative approach was the spiral

model, where development was done in cycles that form an outward expanding spiral (Boehm, 1988). Since then, several agile and iterative approaches have been proposed and adopted into practice (Dima and Mason, 2018). While iterative software development is suitable for most software projects, especially those within an unknown territory for developers or project owners, iterative models have their drawbacks with regard to SDLC management. Agile methods have been criticised for the lack of long-term holistic support, as planning is done in short cycles and may be blind to long term and large scale goals (Baseer et al., 2015; Katayama and Goldman, 2011). To address this issue, frameworks such as scaled agile framework (SAFe) (Turetken et al., 2011) and the scrum of scrums (a scrum meeting of scrum projects) have been developed (Paasivaara et al., 2012). These approaches seek to combine the agility preferred by development teams with more linear or sequential visualisations preferred by upper management.

A recent challenge in the SDLC research has been the shift in the software industry towards continuous development and operations (DevOps) (Ebert et al., 2016; Virmani, 2015). Whereas agile approaches combine the design and development phases, DevOps combines the development and operation phases via utilising continuous integration and delivery to automate and remove extra steps between software development and pushing the system into production (Virmani, 2015). Another change in the field of SDLC research is the microservice approach of building software systems from smaller, sometimes pre-developed, blocks (Ebert et al., 2016; Nagy et al., 2017). This means parts of an IS can be developed individually, ready-made blocks can be utilised, and each part can be conceptualised to follow their own SDLC.

To bring a more concrete look into SDLC models, next we discuss four popular models that can be considered archetypal: waterfall (Balaji and Murugaiyan, 2012), spiral (Boehm, 1988), scrum (Srivastava et al., 2017; Rising and Janoff, 2012), and DevOps (Ebert et al., 2016).

Waterfall: There are multiple versions of the waterfall model (e.g. Royce, 1970; Balaji and Murugaiyan, 2012; Matkovic and Tumbas, 2010). Here we refer to an often cited version as described by Balaji and Murugaiyan (2012). The stages of this model follow each other in a linear, sequential order. The waterfall model is practical due to its simplicity and linear progression. This makes the model easy to follow, and can provide a useful overview of the project. However, this model is criticised for being poor at visualising situations where developers need to return to earlier stages (Matkovic and Tumbas, 2010). To address this issue, some versions of the waterfall model include iterative elements. However, we present the model in its core form, including analysis, design, development, testing, implementation and maintenance stages. The six key stages are described in Figure. 1. Each stage can be seen as containing tasks which the development team needs to take care of. While the model in Figure. 1 is sequential, in practice developers often return to earlier phases.

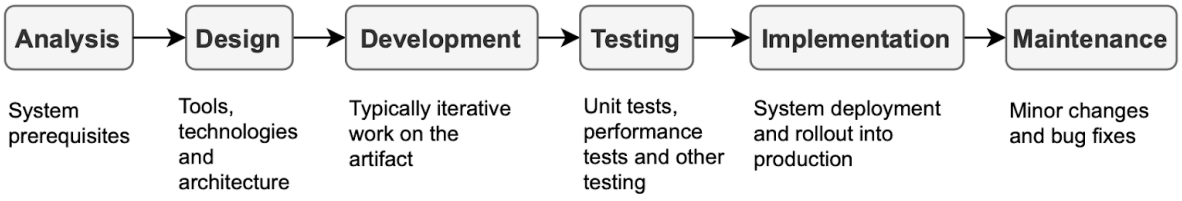

Figure 1. A straightforward waterfall SDLC model

Spiral: To resolve the issues of rigidity and lack of plasticity in the waterfall model, iterative structures were introduced to SDLC models (Matkovic and Tumbas, 2010). One of the first and most cited of these models is the spiral model (Boehm, 1988). Similarly to other SDLC models, there are several variations of the spiral model. Here we present the interpretation of Boehm as described in his seminal work *A Spiral Model of Software Development and Enhancement* (Boehm, 1988). The overall outline of the spiral model is displayed in Figure 2 (left). In this conceptualization, the ISD project iterates through four phases: (1) determining and clarifying objectives; (2) identification of risks and resolving them; (3) developing, coding and testing the solutions; and (4) planning the next iteration. With each new iteration/cycle in the spiral, the cumulative costs of the project increase. At the same

time, the developers and project management continuously learn new information regarding the project. In the planning phase, the project starts out with a rough concept plan, which is then worked into a development plan, and later into a test and operation plan. As an iterative model, feedback from customers, developers and administration can be involved in each cycle - and the development tasks and schedule can be adjusted accordingly (Nilsson and Wilson, 2010).

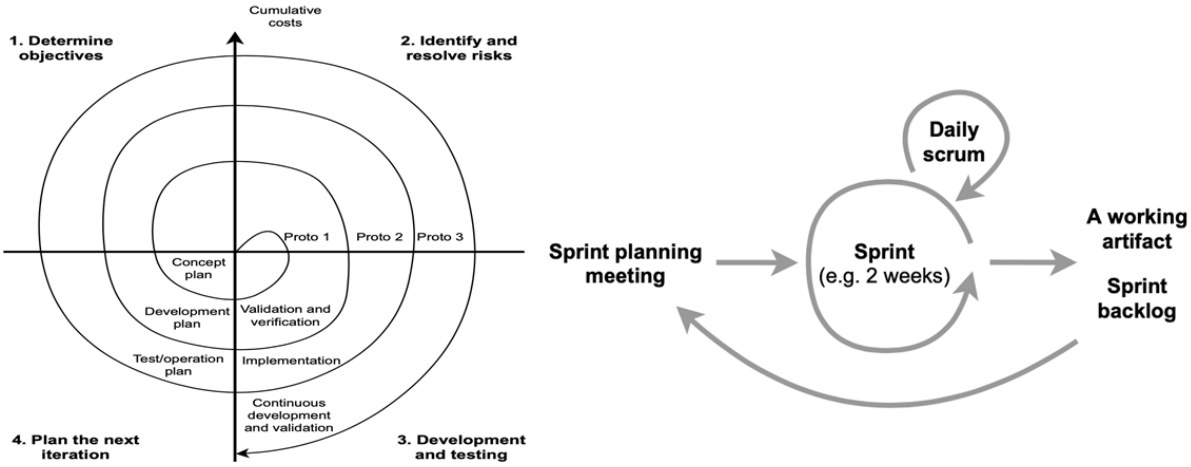

Figure 2. The Spiral SDLC model (left) and scrum (right).

Scrum: Scrum (Figure 2, right) is one the most popular agile frameworks used in ISD. It resembles the spiral model in that software development happens in cycles, which in scrum are called sprints (Srivastava et al., 2017). A single sprint lasts typically somewhere between a week and a month, during which the development team implements a working version of the system they are developing (Rising and Janoff, 2000). The Scrum process starts with sprint planning, which involves aspects such as project goals, selecting tools and looking at the sprint backlog. The team then decides what they are going to work on and proceed to the sprint. A 'standup meeting' takes place each day of the sprint, whereby the project team discusses work in progress and any related concerns (e.g., defects, impediments to flow). At the end of the sprint is a review where the artefact is presented, and new potential tasks are added to the sprint backlog. Testing is done during the sprint, and it is not a phase of its own. During the sprint review, the team can also provide estimations of project completion dates and discuss them with customers and upper management (Rani, 2017).

DevOps: DevOps (Figure 3) is a popular contemporary software development and management paradigm that integrates the previously distinct development and operation stages of software development (Ebert et al., 2016). The goal of DevOps is to deliver software products to customers as quickly as possible, to avoid manual needless labour and to guide software development towards efficient practices (Ebert et al., 2016, Virmani, 2015). The success that big tech companies such as Google and Amazon have had with the approach has accelerated its popularity and adoption in smaller businesses and a wider range of projects and products (Ebert et al., 2016). DevOps builds on the concept of a continuous delivery pipeline, where building, testing, quality assurance, verification and development are all automated (Virmani, 2015). However, the rapid deployment process of DevOps can make the approach unsuitable to be applied in systems with high security requirements (Ebert et al., 2016).

The DevOps process is cyclical, but in each cycle there are sequential steps that are taken. Development work of individual components is continuously integrated to the full system and shared between other contributors (Virmani, 2015). The build goes through a set of tests and quality monitoring, all of which are automated as far as possible. Feedback and data from system use is collected via monitoring tools. Based on this, constant planning takes place that guides the

development of the system (Virmani, 2015). It is worth noting that recently a popular re-contextualization of DevOps in the context of ML development (coined MLOps) has emerged (Karamitsos et al., 2020). MLOps intends to bridge design, model development and operations, and apply DevOps principles to ML development. In practice, similarly to DevOps, MLOps focuses on building automated pipelines that streamline processes involved in ML model development including data extraction and curation, model training, continuous testing and monitoring. While MLOps is a promising development from the perspective of integrating ML in SDLC models, at this paper's level of abstraction, we focus on broad challenges in an attempt to also explain the reasons why MLOps has proliferated among data scientists and ML engineers (Valohai, 2021).

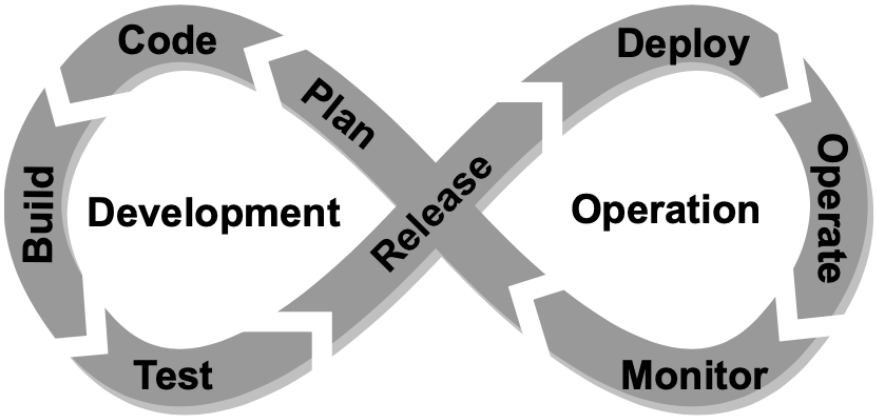

Figure 3. A popular hourglass visualisation of DevOps.

2.2 Machine learning in software development

ML is a broad term that encompasses various computer-based data-mining and interpretation techniques used for uncovering complex patterns, particularly in large and complex sets of data (Mitchell, 1997; Mohri et al., 2018; Shalev-Shwartz and Ben-David, 2014) to extract insights for classification, prediction, and decision-making purposes (e.g., Chinnamgari, 2019; Cui et al., 2006). ML can be divided into subcategories such as supervised; unsupervised; semi-supervised; transfer; and reinforcement; learning (Chinnamgari, 2019). Today, ML is applied to solve a wide variety of real world problems, for example, in finance (Henrique et al., 2019) and cybersecurity (Laato et al., 2020; Salloum et al., 2020), and depending on the specific use case, some ML approaches work better than others (Caroline Cynthia and Thomas George, 2021). Depending on the business case, the ML development may involve a great deal of data curation, annotation and experimentation, while other cases may be more straightforward. In general, ML system development is distinctively different from "standard" software engineering, due to, for example, the importance of data and the models being probabilistic as opposed to deterministic (Akkiraju et al., 2020; Ishikawa and Yoshioka, 2019).

The two key concepts of this paper, ML and SDLC models, are both simplified abstractions of complex real-world phenomena (Dignum, 2019). The growing popularity of ML development in ISD brings new characteristics to the development practices and the developed systems which may not yet be reflected in existing SDLC models. Three novel features of ML systems may be highlighted: *Autonomy*, referring to how ML systems may make decisions on how to operate in their environment independently from human guidance, *interactivity*, referring to the increased reciprocal interaction of ML systems with their environments, and *adaptability*, meaning that if the ML system is updated with new models trained with new data, the system as a whole displays characteristics of learning from its environment (Dignum, 2020). The inscrutability of, in particular more complex deep neural networks, means understanding them is often only possible indirectly via test data sets, or sometimes via explainable AI techniques (Asatiani et al., 2021; Holstein et al., 2019).

Data scientists and ML engineers are key actors in ML development. The most commonly associated activities of these roles such as working with data, feature engineering, tweaking model parameters, creating test data sets and so forth are largely different from standard programming more typically done by software engineers (Jüngling et al., 2020). For this reason, research is needed on whether and how existing popular SDLC models support and can accommodate ML model development work (ibid). Looking at the four archetypal SDLC models discussed in Section 2.1, the more linear models may be of limited benefit in describing the sometimes experimental and iterative work of data scientists. By contrast agile approaches such as scrum may provide inadequate support for management to control and understand the ML development work. While such general statements can be made, more detailed insights into the interplay between conceptual SDLC models and ML development are needed.

3 Methodology

3.1 Data collection

We chose an expert interview approach to solicit expert views on how ML development work should be integrated into SDLC models. For guiding the interviews, we used the four archetypal SDLC models reviewed in Section 2.1 as a basis for the discussion. While recent literature suggests most contemporary ML system development follows the MLOps paradigm (Tamburri, 2020; Valohai, 2021), we felt that asking the experts to reflect on how ML fits into other popular conceptual SDLC models would produce rich data that could serve as a base for more nuanced and even surprising discoveries. We selected the four SDLC frameworks based on the criteria of prominence and popularity, and they were the following: waterfall (Balaji and Murugaiyan, 2012), spiral (Boehm, 1988), scrum (Srivastava et al., 2017; Rising and Janoff, 2012), and DevOps (Ebert et al., 2016). While waterfall and spiral models are rarely used by developers themselves, they remain preferred tools for software project management (Dima and Maassen, 2018; Niederman, 2021). By contrast, agile and DevOps approaches are widely used in contemporary software development.

We employed ‘purposeful sampling’ (Seidman, 2019) to recruit practitioners with experience in AI technologies and SDLCs. As a recruitment criteria, the participants had to have at least 5 years of industry experience working closely with either ML engineering or SDLC management. We began by reaching out to three academic experts with on-going industry collaboration, and asked them to provide and suggest names for interviews. We then asked the suggested experts to be interviewed and for them to suggest further names. We performed interviews until we felt that no new significant or contradicting information had emerged during the last three interviews, resulting in the final number of 19 interviews (see Table 1). The interviews were conducted remotely via Zoom or Teams by the first author. The duration of interviews ranged between 45 and 115 minutes. Interview notes were taken during and immediately after the interviews, and all interviews were recorded and transcribed.

ID	Job title	Sector type	Years of industry experience
1	Data & security specialist	Public	5
2	Chief technology officer	Private	11
3	Professor of data science	Public	16
4	Competence lead	Private	10
5	Senior data scientist	Private	20
6	Senior software developer	Private	13
7	Data science research fellow	Public	15
8	Director of AI-startup	Private	20
9	Professor of software engineering	Private	20

10	Systems architect	Private	6
11	Actuary	Private	7
12	Professor of software engineering	Public	26
13	Professor of data science	Public	29
14	Director of software development	Private	22
15	Senior software developer	Private	26
16	AI consultant expert	Private	11
17	Data project lead	Private	12
18	Director of AI-startup	Private	15
19	Professor of software engineering	Public	21

Table 1. Profile of interviewees

The interviews were conducted as semi-structured interviews (Coombes et al., 2009). Interviewees were presented one SDLC model at a time, and the same set of thematic questions was asked regarding each model: (1) first-hand experiences with the model; (2) evaluation of the positive and negative aspects of the model; and (3) the ability of the model to meaningfully describe ML model development work and the characteristics of ML systems. Each of these three questions was followed by a set of clarifying questions adapted according to the interviewees' responses. The interviews primarily focused on the third theme: the suitability of the models to describe ML system development. After going through the four SDLC models, the interviewees were asked to synthesise their thoughts by comparing the four presented models with each other and to evaluate which of the four models best accommodates ML model development and why.

With regard to the four discussed models (e.g., waterfall, spiral, scrum, and DevOps), the interviewees had varying experiences and knowledge. All participants were familiar to some degree with waterfall, scrum and DevOps, while spiral was less known. Larger differences were exhibited in the participants' knowledge and experience concerning ML engineering. The variance in participants' backgrounds could be considered an advantage, as it produced rich and varied data During the interviews, the participants discussed various strengths and weaknesses of the models, in particular with regards to ML development. They supported their arguments with real-world examples from projects and business cases in which they had been involved.

3.2 Data analysis

We approached data analysis inductively, guided by the RQ: *what are the challenges related to integrating ML systems with SDLC models?* As is typical with inductive qualitative analysis, we began the analysis by familiarising ourselves with the interview material. This was done by listening to the interview recordings and reading through the notes and transcriptions. During this stage we made additional notes about interesting observations, remarks and our own emerging thoughts related to the RQ. In the second step, we coded the transcribed data to mark concrete challenges and issues in integrating ML system development into conceptual SDLC models. The coding was performed by the first author. Here we already attempted to move beyond the four archetypal models and sought to identify more universal challenges as opposed to those specific to a certain conceptualisation. These codes described issues mentioned by the participants such as the lack of shared vision between (1) development teams; and (2) management or customers; and that the role of a data scientist was not always easy to define.

In the third step of the analysis we took the coded basic level ideas and collated them into broader level themes (Braun and Clarke, 2006). All authors participated in this process. We began from the codes identified in the previous step and proposed various conceptualisations that would best describe the data. Through iteration and discussions we arrived at three large themes, which all had interplay

with one another and which can be considered to represent areas of focus when merging ML development with SDLC conceptualisations.

4 Findings

The thematic analysis led to the discovery of findings in three broad themes: (1) the redefinition of prescribed roles and responsibilities in SDLC models, (2) the use of MLOps as a frame for creating a shared understanding and commitment for management and software teams, and (3) a shift from adherence-based SDLC approaches towards value-based approaches (see Figure 4). We present and discuss our findings within each of these three high-level themes below.

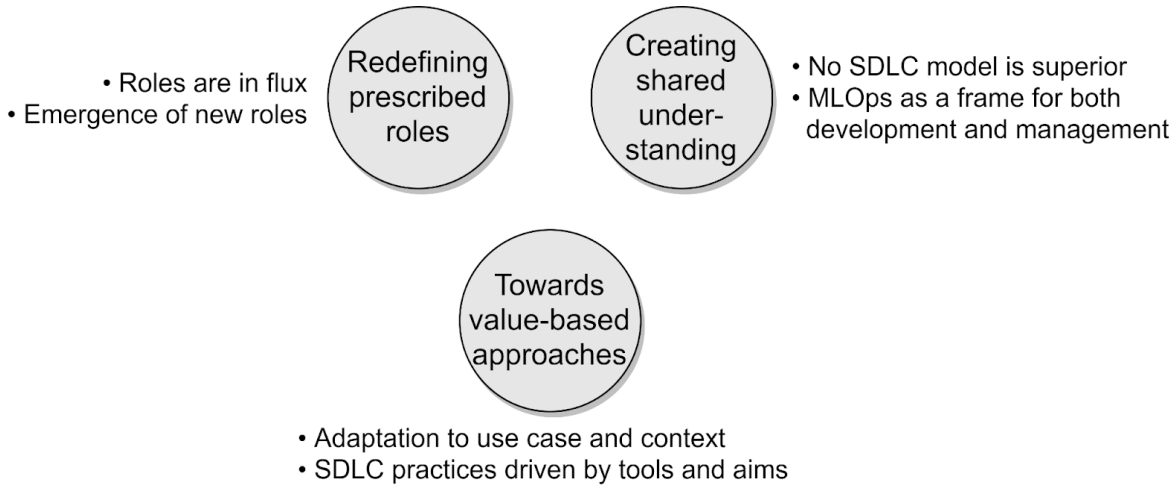

Figure 4. The three key themes and related content identified through thematic analysis.

4.1 Redefining the prescribed roles and responsibilities in SDLCs

The first theme involves the redefinition of roles and responsibilities related to SDLCs. Most of the interviewees involved in technically developing ML systems reported that they mostly work alongside other engineers and follow the SDLC model of the entire development team. For example, with scrum, their sprint tasks could be quite different, but they would follow the same sprint and meeting schedules. A data scientist could also get support for some aspects of their work from the other team members. While the role of a data scientist was singled out by the informants, several participants implied that the roles in development teams can be fluid depending on the task at hand and team composition. Even in the primary role of a data scientist, the activities could vary considerably between projects, as the following quotation illustrates:

"I've been doing everything [from data extraction to model training and production]. But sometimes we have distinct roles. For example, we had a case where I was together with another data scientist, I was the person who pumped information from the clients and took care of the background stuff while the other guy worked more on data processing and that end." (P5)

Despite drawing a parallel between specialised development roles and the role of a data scientist, some interviewees brought up AI-specific SDLC models, and data mining process models such as CRISP-DM and the Essence theory of software engineering. An interviewed professor of software engineering was particularly critical about the idea of there ever being a model that would fit all or even most software development cases and even challenged the idea of advocating for any specific SDLC model:

"I've tried to understand for years why we are still going to this direction where someone presents yet another SDLC model and again we hear cheering for it from the advocates of the new model. There are always unique structures and organisations and there will always be the need to fit and apply SDLC models into that context." (P12)

The interviewees also brought up aspects that indicate that there is an ongoing convergence between the roles of data scientists and software engineers. For example, the interviewed director of software development (P14) stated that as the processes of creating certain standard ML systems are increasing in maturity, they are simultaneously becoming a part of the toolkit of regular software developers. Similarly, data scientists are adopting the same tools (e.g., git for version control and CI/CD pipelines) as software developers. The interviewed actuary (P11) stated that they use many professional software development tools to integrate their work into the company's IT systems. Moving beyond these examples, several of the interviewees argued that knowledge and understanding of data, in general, is becoming a core requirement not only for data scientists, but also for software developers and upper management.

Furthermore, analysis of the interview data suggests that integration of ML engineering into software development is piling pressure on software developers to personally develop their skill set to learn how to manage large data sets and train basic ML models. The Data Project Lead (P17) was an advocate for the need for all engineers to learn the basics of ML model development:

"Let's say it this way that everyone should know PyTorch and the basic algorithms that when they have a challenge, and they know even a little bit about these techniques, they start seeing situations where [ML] can be applied. And of course it should not be thrust in there by force if no value is added, but in my opinion it should be in control. And with this I mean an engineer approach of applying with PyTorch, TensorFlow or whatever, ML for solving a specific solution." (P17)

Taken together, the broader theme arising from the interviews with regards to the presubscribed roles in software development is that they appear to be in flux. The proliferation of data-intensive development is pressuring upper management to acquire an understanding of how to best make use of new opportunities, as well as how the development of ML systems and their SDLC should be managed. Simultaneously data scientists are in the process of increasing the level of maturity of their activities, as challenges such as AI governance and auditing, model biases and interpretability of the models seem pertinent in most AI systems. In this complex, constantly transforming landscape, new software roles are likely to emerge, formed and shaped by new emerging technologies such as new ML approaches, digital platforms and the overall surrounding digital infrastructure.

4.2 MLOps as a frame for creating a shared understanding and commitment by management and software teams

The second theme relates to the creation of shared understanding and commitment among management and software project teams, and particularly the role of the recent MLOps model in this process. Here we provide a synthesis of the interviews regarding each of the archetypal SDLC models, followed up by why the interviewees in general picked MLOps as the most potent approach to ML system development. However, the interviewees did not consider any of the archetypal SDLC models to be obsolete. Individual interviewees expressed views in support of the spiral model, waterfall as well as the more recent agile approaches, DevOps and MLOps.

The waterfall model for ML engineering: Initially the researchers were surprised that a couple of participants favoured the waterfall model over alternatives. The interviewed data science research fellow argued for the usefulness of the waterfall model as follows:

"Of course I should not probably say this, but the waterfall model is the most readable. You have to remember I've not been involved in very big software projects or the industry, so from that point of view I don't have knowledge of how to get a project of 100 people to actually work. (–) But I would say that instead of slavishly following any of the presented models, I prefer knowing the steps involved in the development process." (P7)

Interviewees reported several positive and negative aspects concerning the waterfall model that were more universal than just related to ML development, such as the waterfall model being good for projects where the developers knew exactly what they were going to do beforehand. Interviewees agreed that the waterfall visualisation is the simplest one and, therefore, can be used to quickly obtain

an overview of the development work. It may be adopted to provide a brief overview of what needs to be done and at what stages. When asked about the model's suitability for ML engineering in particular, the following quotation from a professor of data science illustrates the majority opinion particularly well:

"None of these is superior and I'm sure the waterfall is also a good model if you have a very accurate picture about what you are doing, but of course I have never really felt like I would have, unless the problem is really simple." (P3)

The Spiral model for ML engineering: One of the main criticisms towards the spiral model was the fixed size of quadrants, and that this issue was not easily solvable, as adjusting the quadrants to match the workload would make the conceptualization unreasonably confusing. However, interviewees argued that such an iterative structure could actually be quite suitable for supporting ML development. For example, a data scientist working for a private software consulting company said the following:

"I like the simple approach and the engineer-like clarity [of the spiral model]. (–) In a certain way the cyclical work is presented well here, as if peeling an onion one layer at a time, or rather growing an onion one layer at a time. It is a good description of ML model development." (P5)

Scrum for ML engineering: Some of the interviewees challenged the conception of viewing scrum as an SDLC model. Instead, they considered it as *"A way of doing software"*. None of the interviewees denied the usefulness of iterative development, but emphasised that scrum was not the only available iterative model and referred to other agile practices as potential alternatives. Related to scrum being a way of working, interviewees said they were using scrum together with DevOps in their work, and also saw this as the most promising approach when working towards a potentially new SDLC model that would accommodate the characteristics of ML model development.

DevOps for ML engineering: The principles of continuous integration and delivery (CI/CD) in DevOps were seen to align with ML model engineering, as reported in extant literature (e.g., Karamitsos et al., 2020). Most of the interviewees, particularly those working in the industry, stated that almost all their work these days follows DevOps. However, DevOps has gained popularity independently of the rise of ML model engineering. With regards to DevOps, participants sometimes discussed it with the concept of MLOps interchangeably. Interviewees appreciated how the operation-side was taken into account in DevOps/MLOps, and considered it essential in ML development. The following quotation highlights this:

"From the viewpoint of my work, in the real world that constantly changes, the model here that shows a DevOps- approach would be best. (–) When the system is in production we can follow it, and from there we can get a trigger that we now need to react and do something." (P11)

The proliferation and use of MLOps: Moving beyond the individual models, one of the issues often brought up in the interviews was the lack of understanding between software engineers, data scientists, customers and project management. In particular, informants who held developer positions, argued that managers who do not understand the basics of the technologies involved in a project are in no place to lead the project. When synthesising the arguments across all the interviews, MLOps emerged as a promising SDLC frame which could simultaneously work as the backbone of development, but also as a conceptualization for the upper management on the state of the IS. Going even further, the interviewed data project lead (P17) spoke for the importance of integrating governance and automated testing into MLOps pipelines, hence automating the roles of external auditors and managers to an extent. In this way, MLOps has the potential to bridge the chasm between management, customers, data scientists and software development teams.

4.3 ML and SDLC method tailoring

The third theme can be summarised as a shift from adherence-based SDLC approaches towards method tailoring (Campanelli and Parreiras, 2015; Dingsøyr et al., 2019). One of the challenges that software development teams encounter is the strict requirement to adhere to the SDLC approach (e.g., daily standup, two-week sprint, retrospective) as prescribed by the method creator, without

considering the socially embedded contextual nature of software development practices. However, the interview data suggests that development teams are in practice quite liberal in applying the theoretical models in practice, meaning they adapt each of the models for the specific use case and context. For example, a senior software developer working for a large software consulting company (P6) reported that *"they always follow some iterative development approach"*, preferably Scrum, but that their clients may not have the sufficient level of maturity nor willingness to participate in the Scrum process. Hence software development teams aim adjust their SDLC approach to accommodate the customers' capability and needs, but this is not always straightforward. Issues arise when technologies and development practices advance faster than customers' understanding and maturity, as indicated by the quote below:

"As I said earlier, the development of product management is highlighted significantly – wheather you are doing a service or software products, or these kinds of systems that we have. And it is also challenging from the business viewpoint to implement – you can imagine that the [redacted] company that is used to buying our [product], they expect us to deliver the product that is mostly hardware. And when we say that there's now software involved as well and you should even pay extra for it, they are not ready at all. It's a change, but we just have to go with it (laughs)." (P14)

According to the interviewees, ML engineering consists of various activities that require specialized skills, most important of which are data collection and data engineering, as well as model development and testing. Particularly among larger companies, the work of ML engineers may also consist of providing a pipeline where the latest champion model is provided for use to the rest of the system development team. The interviewed data project lead (P17) illustrates this with an example from their company where two data scientists work relatively independently from the rest of the team during Scrum sprints, but would always seek to integrate their work to the rest of the product at the end of the sprint. In doing so, the data scientists were subjected to the same SDLC procedures as the rest of the team. The ML development practices are hence being guided by specific development tools and aims, and may follow a rigid structure despite not following any specific SDLC model:

"We have a clear setup, time windows, tasks and everything - we provide our work to the pipeline where they move onward and are integrated in the system. (...) Of course there's a lot of tests and testing involved." (P11)

As ML engineers and data scientists are already working in development teams following established SDLC models, and furthermore, software engineers are feeling the pressure to learn the basics of ML model development, it is clear that ML model development has already been integrated into existing SDLC models at the level of practice. The requirement to bring ML under a strict SDLC process involved aspects such as performing governance activities, audits, rigorous testing and consequently improving the level of maturity of the ML system development processes and making ML development a standard and basic part of the overall ISD. The following quote illustrates this:

"The [company we're working with] have this nice saying that they want to make AI development boring. I like that. There was this mystification around AI – it was a hype technology but no one really understood what it was about. We're now at a stage where companies are really seriously building AI tools and systems that they actually use and now suddenly we have to think about compliance, governance, auditing and so forth." (P8)

With the arrival of MLOps, construction of CI/CD pipelines and use of version control tools such as Delta Lake, it appears that there is a shift in software development that goes beyond the integration of ML engineering into SDLCs, that overall can be characterised as a shift towards value-based approaches. The on-going integration of the work of data scientists to the rest of the software development team is currently largely done at the level of individual projects based on their needs, but there are also conceptual advances from the direction of cloud service providers and academia. In the future we may see increasing influence of the cloud service providers on the development work as suggested by the following quotes:

"[Cloud services] are pushing - marketing their own APIs and management solutions. There is TensorFlow and it uses Google's TPUs and it is a whole ecosystem that hugely helps in [ML] development." (P5)

"It's a value question perhaps that what kinds of software is useful to be in the hands of the company and what should be acquired from elsewhere. Let's say that you start doing everything bottom-up, all the lower end libraries and frameworks from the start, then that is defininently a strategic decision for the company. I mean if you start doing fundamental things yourself, it has to provide significant added value to the customer (…). I see that the treshold from this kind of from scratch development work is only higher and higher. It's getting more and more difficult to justify a decision not to, for example, utilize cloud platforms." (P2)

5 Discussion

Our results contribute broadly to the IS literature on SDLC management (e.g., Dennehy and Conboy, 2018; Griva et al., 2020). On a general level, the findings from the interviews suggest that ML development is becoming increasingly integrated into ISD practices. The results underscore the fact that there are different types of unique characteristics (e.g. probabilistic software, inscrutability of the models) and contextual factors (the huge demand for high quality training data) involved in basically all ML system development projects. Similar to the rest of ISD, it is highly unlikely that a single SDLC will turn out to be the only feasible conceptualization for guiding ML system development. The interviewees highlighted that the archetypal SDLC models are significantly modified when used in practice, with company culture, project needs and skillsets of the stakeholders all influencing which approach to take. Altogether, there was a consensus among interviewees that there were use cases for each of the four presented SDLC models in ML system development. However, this being said, the MLOps approach showed promise in serving as a conceptualization that can bring both developers and management to have a shared, unified understanding of the ISD process.

The current study makes three principal contributions to the body of literature on the integration of ML engineering into conceptual SDLC models (Jüngling et al., 2020; Amershi et al., 2019). First, our findings suggest that there is an ongoing redefinition of prescribed developer roles with regards to ISD involving ML. The work of data scientists is being integrated into various IS, including also high-risk systems. This transition forces data scientists to adopt practices from software engineering and integrate their work with ISD practices. In this complex process, the prescribed roles of software engineers and data scientists are converging, and possibly new roles are created, such as an ombudsman responsible for AI governance (e.g., Floridi et al., 2018). Second, our findings highlight MLOps as a unifying frame for creating a shared understanding and commitment for management and software teams. MLOps is based on constructing CI/CD pipelines and on automating the creation of ML systems, with the aim of making ML system development more mature (Valohai, 2021). MLOps is said to have been born out of the necessity to streamline and guide ML engineering to become suitable for industry use (Tamburri, 2020). Our results underscore that instead of relying on simplified descriptions of the development process, management should ensure they understand what the developers are doing and use the same frame as the developers (e.g. MLOps) for managing ML development projects. Third, our findings support previous findings (e.g., Giardino et al., 2015; Kemell et al., 2018) that no single SDLC model is universally superior due to differences in software projects, aims, goals, organisational context, team profiles, and responding to continuous change requests from customers. However, our findings also indicate that not all SDLCs are of equal use, and the ISD in general is moving away from adherence-based SDLC approaches towards value-based approaches – of which a good example is the proliferation of the DevOps/MLOps paradigms.

Our findings also have practical implications. First, they suggest that the SDLC model that guides development work may act as a shared frame for the development team, management and customers. For this to happen, management and customers need to accrue knowledge of both SDLC models and ML development. Second, we found evidence that there was convergence between prescribed developer roles within ML development teams in that software engineers feel pressure to learn to

utilise basic ML APIs in their development work, and similarly data scientists are pressured to be able to integrate their work and the models they produce as part of the larger IS. Third, the interviewed informants seemed almost unanimous in that no single SDLC would be suitable for all projects and purposes. Hence, purely adherence-based development practices may be even less frequent in the near future.

As with all research, however, we acknowledge this study has some limitations. In particular, we point out two limitations, which also provide opportunities for future research. The first relates to our interview methodology, where we drew knowledge from experts in either ML systems or SDLC models. While with our sampling we reached a saturation, complementary viewpoints may be obtained through other approaches such as case studies. Due to the nature of the data we did not perform checks for inter-rater reliability and the rich data left room for other interpretations. For these reasons we encourage future research to continue this line of inquiry. In particular we see value in involving theoretical lenses in the analysis to better ground the study to existing knowledge frameworks. The second limitation relates to our approach that was grounded on the SDLC research tradition, which inevitably directed the focus of the empirical inquiry. Future research could advance understanding of the role of the external factors (e.g., national culture, regulated environments) that could help developers better account for possible external obligations and demands that ultimately affect the ISD activities. Future research could also examine the ML system development process from one abstraction level higher than the SDLC to better account for the influence of organisational and environmental factors on the ML system development process.

6 Conclusion

The proliferation of ML systems, among other changes in the software development industry, has implications on popular SDLC models, and this transition is ongoing. Our findings to the RQ: "*What are the challenges related to integrating ML systems with SDLC models from ISD practitioners' vantage point?"* highlight the heterogeneity of needs and contextual factors with respect to ML development as well as the accommodation and management of ML systems. This, in turn, creates a need for tools and conceptual support to manage the life cycle of ML systems. Our results highlight the variability in ML system development and the importance of flexibility in choosing the approaches to support the development activities. The proliferation of data-intensive development and probabilistic systems pose challenges not only for developers, but also for the management. To alleviate issues pertaining to the lack of understanding between developers and management, our findings highlight MLOps as a promising conceptualization that helps all stakeholders involved in the ISD acquire a shared vision.

Our findings support the interpretation that ML development is increasingly converging with contemporary software development. ML systems are being developed according to similar principles that have already been established in software system production. For ML system development, SDLC models continue to possess largely the same benefits and drawbacks as with any system development. We conclude that contemporary SDLC models seem to effectively accommodate ML system development, but we also note that adjustments have been made in the industry as indicated by the conceptual transformation of the DevOps paradigm into MLOps.

References

Akinsola, J. E., A. S. Ogunbanwo, O. J. Okesola, I. J. Odun-Ayo, F. D. Ayegbusi, and A. A. Adebiyi (2020). "Comparative Analysis of Software Development Life Cycle Models (SDLC)." in *Computer Science Online Conference*.

Akkiraju, R., Sinha, V., Xu, A., Mahmud, J., Gundecha, P., Liu, Z, Xiaotong Liu and Schumacher, J. (2020). "Characterizing machine learning processes: a maturity framework." in Fahland D., Ghidini C., Becker J., & Dumas M. (eds.) *International Conference on Business Process Management*, Seville, Spain.

Amershi, S., A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann (2019). "Software Engineering for Machine Learning: A Case Study." in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice. ICSE-SEIP'19*. Montreal, Quebec, Canada.

Asatiani, A., P. Malo, P. R. Nagbøl, E. Penttinen, T. Rinta-Kahila, and A. Salovaara (2021). "Sociotechnical Envelopment of Artificial Intelligence: An Approach to Organizational Deployment of Inscrutable Artificial Intelligence Systems." *Journal of the Association for Information Systems* 22 (2), 8.

Balaji, S. and M. S. Murugaiyan (2012). "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC." *International Journal of Information Technology and Business Management* 2 (1), 26–30.

Baseer, K., A,. R. M. Reddy, and C. S. Bindu (2015). "A systematic survey on waterfall vs. agile vs. leanprocess paradigms." *i-Manager's Journal on Software Engineering* 9 (3), 34.

Bawack, R.E. and Ahmad, M.O. (2021). "Understanding business analytics continuance in agile information system development projects: an expectation-confirmation perspective", *Information Technology & People*, Vol. ahead-of-print No. ahead-of-print.

Banerjee, S., Singh, J.P., Dwivedi, Y.K. and Rana, N.P. (2021). "Social media analytics for end-users' expectation management in information systems development projects", Information Technology & People, Vol. ahead-of-print No. ahead-of-print.

Boehm, B. W. (1988). "A spiral model of software development and enhancement.", *Computer* 21 (5), 61–72.

Braun, V. and V. Clarke (2006). "Using Thematic Analysis in Psychology." *Qualitative Research in Psychology* 3 (2), 77–101.

Campanelli, A.S. and Parreiras, F.S. (2015). "Agile methods tailoring–A systematic literature review", *Journal of Systems and Software*, (110), 85-100.

Caroline Cynthia, P., & Thomas George, S. (2021). An outlier detection approach on credit card fraud detection using machine learning: a comparative analysis on supervised and unsupervised learning. In Intelligence in Big Data Technologies—Beyond the Hype (pp. 125-135). Springer, Singapore.

Chinnamgari, S. K. (2019). *R Machine Learning Projects: Implement supervised, unsupervised, and reinforcement learning techniques using R 3.5*. 1st Edition. Birningham, UK.

Coombes, L., D. Allen, D. Humphrey, and J. Neale (2009). "In-depth interviews." *Research methods forhealth and social care*, 197–210.

Cui, G., M. L. Wong, and H.-K. Lui (2006). "Machine Learning for Direct Marketing Response Models: Bayesian Networks with Evolutionary Programming." *Management Science.*

Dennehy, D. and Conboy, K., (2018). "Identifying challenges and a research agenda for flow in software project management", *Project Management Journal*, *49* (6), 103-118.

Dennehy, D. and Conboy, K. (2019). "Breaking the flow: a study of contradictions in information systems development (ISD)", *Information Technology & People*, *33* (2), 477-501.

Dignum, V. (2019). "Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way." Springer International Publishing.

Dignum, V. (2020). "Responsibility and Artificial Intelligence." *Oxford Handbook of Ethics of AI*. Oxford, Oxford University Press,

Dima, A. M. and M. A. Maassen. (2018). "From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management." *Journal of International Studies* 11 (2), 315–326.

Dingsøyr, T., Dybå, T., Gjertsen, M., Jacobsen, A. O., Mathisen, T. E., Nordfjord, J. O., ... & Strand, K. (2019). "Key lessons from tailoring agile methods for large-scale software development", *IT Professional*, *21*(1), 34-41.

Ebert, C., G. Gallardo, J. Hernantes, and N. Serrano (2016). "DevOps." *IEEE Software* 33 (3), 94–100.

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., et al. (2018). "AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations." *Minds and Machines, 28* (4), 689–707.

Giardino, C., N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson (2015). "Software development in startup companies: the greenfield startup model." *IEEE Transactions on Software Engineering* 42 (6), 585–604.

Griva, A., Byrne, S., Dennehy, D., and Conboy, K. (2020). "Software Requirements Quality: Using Analytics to Challenge Assumptions at Intel," *IEEE Software*.

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. Expert Systems with Applications, 124, 226-251.

Holstein, K., J. Wortman Vaughan, H. Daumé, M. Dudik, and H. Wallach (2019). "Improving Fairness in Machine Learning Systems: What Do Industry Practitioners Need?" in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. CHI '19.* New York, NY, USA

Ishikawa, F. and N. Yoshioka (2019). "How do engineers perceive difficulties in engineering of machine learning systems? Questionnaire survey." *in 2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP).*

Jüngling, S., M. Peraic, and A. Martin (2020). "Towards AI-based Solutions in the System Development -Lifecycle." in *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering.*

Laato, S., Farooq, A., Tenhunen, H., Pitkamaki, T., Hakkala, A., & Airola, A. (2020). AI in cybersecurity education-a systematic literature review of studies on cybersecurity moocs. In 2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT) (pp. 6-10). IEEE.

Laato, S., Mäntymäki. M., Islam, A.K.M.N., Hyrynsalmi, S. & Birkstedt, T. (2022) "Trends and Trajectories in the Software Industry: Implications for the Future of Work", *Information Systems Frontiers.*

Lyytinen, K. and Rose, G. M. (2006). "Information System Development Agility as Organizational Learning," *European Journal of Information Systems*, 15:2, 183-199.

Karamitsos, I., S. Albarhami, and C. Apostolopoulos (2020). "Applying DevOps practices of continuous automation for machine learning." *Information* 11 (7), 363.

Katayama, E. T. and A. Goldman (2011). "From manufacture to software development: a comparative review." in *International Conference on Agile Software Development*.

Kemell, K.-K., A. Nguyen-Duc, X. Wang, J. Risku, and P. Abrahamsson (2018). "The essence theory of software engineering–large-scale classroom experiences from 450+ software engineering BSc students." in *International Conference on Product-Focused Software Process Improvement*.

Mantei, M. M. and T. J. Teorey (1989). "Incorporating behavioral techniques into the systems development life cycle." *MIS Quarterly*, 257–274.

Matkovi´c, P. and P. Tumbas (2010). "A comparative overview of the evolution of software development models." *International Journal of Industrial Engineering and Management* 1 (4), 163–172.

Minkkinen, M., Niukkanen, A., & Mäntymäki, M. (2022). What about investors? ESG analyses as tools for ethics-based AI auditing. AI & Society, 1-15.

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill Series in Computer Science. New York: McGraw-Hill.

Mohri, M., A. Rostamizadeh, and A. Talwalkar (2018). Foundations of Machine Learning. Second edition. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press.

Mäntymäki, M., Minkkinen, M., Birkstedt, T. et al. Defining organizational AI governance. *AI and Ethic*s (2022). https://doi.org/10.1007/s43681-022-00143-x

Nagy, D., L. Schultz, and T. Wiederker (2017). "Replace or revise? A case study investigating the replacement of an organizational website." in *AMCIS2017 and eBusiness and eCommerce Digital Commerce (SIGeBIZ).*

Niederman, F. (2021). "Project management: openings for disruption from AI and advanced analytics", *Information Technology & People*, Vol. ahead-of-print No. ahead-of-print.

Nilsson, A. and T. L. Wilson (2012). "Reflections on Barry W. Boehm's "A spiral model of software development and enhancement"." *International Journal of Managing Projects in Business.*

Paasivaara, M., C. Lassenius, and V. T. Heikkilä (2012). "Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?" in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*.

Ragunath, P., S. Velmourougan, P. Davachelvan, S. Kayalvizhi, and R. Ravimohan (2010). "Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC)." *International Journal of Computer Science and Network Security* 10 (1), 112–119.

Raji, I. D., A. Smart, R. N. White, M. Mitchell, T. Gebru, B. Hutchinson, J. Smith-Loud, D. Theron, and P. Barnes (2020). "Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing." in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. FAT* '20.* New York, NY, USA

Rani, S. B. A. S. U. (2017). "A detailed study of Software Development Life Cycle (SDLC) models." *International Journal of Engineering And Computer Science* 6 (7).

Rising, L. and N. S. Janoff (2000). "The Scrum software development process for small teams." *IEEE software* 17 (4), 26–32.

Royce, W. (1970). "Managing the development of large software systems." in *Proceedings of the 9th international conference on Software Engineering*.

Rubin, M. L. (1970). "Introduction to the system life cycle." *Handbook of Data Processing Management*. Vol. 1.

Salloum, S. A., Alshurideh, M., Elnagar, A., & Shaalan, K. (2020). Machine learning and deep learning techniques for cybersecurity: a review. In The International Conference on Artificial Intelligence and Computer Vision (pp. 50-57). Springer, Cham.

Schlesinger, A., O'Hara, K. P., & Taylor, A. S. (2018). Let's talk about race: Identity, chatbots, and AI. In Proceedings of the 2018 chi conference on human factors in computing systems (pp. 1-14).

Seidman, I. (2019). *Interviewing as Qualitative Research: A Guide for Researchers in Education and the Social Sciences.* New York, NY: Teachers College Press.

Shalev-Shwartz, S. and S. Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms.* New York, NY, USA: Cambridge University Press.

Srivastava, A., S. Bhardwaj, and S. Saraswat (2017). "SCRUM model for agile methodology." in *2017 International Conference on Computing, Communication and Automation (ICCCA). IEEE*.

Studer, S., T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Müller (2021) "Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology." *Machine Learning and Knowledge Extraction* 3 (2), 392–413.

Tamburri, D. A. (2020). "Sustainable MLOps: Trends and Challenges." in *22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC).* IEEE, pp. 17–23.

Turetken, O., I. Stojanov, and J. J. Trienekens (2017). "Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework." *Journal of Software: Evolution and processs* 29 (6), e1796.

Valohai (2021). *Practical MLOps: How to Get Ready for Production Models. ebook* URL: https://valohai.com/mlops-ebook/ (visited on August 18th, 2021)

Virmani, M. (2015). "Understanding DevOps & bridging the gap from continuous integration to continuous delivery." in *Fifth international conference on the innovative computing technology* Galcia, Spain.