It Takes Three to Tango: Requirement, Outcome/data, and AI Driven Development

Jan Bosch[1], Helena H. Olsson[2] and Ivica Crnkovic[1]

[1]Chalmers University of Technology, Department of Computer Science & Engineering, Göteborg, Sweden
[2]Malmö University, Department of Computer Science and Media Technology, Malmö, Sweden

Abstract. Today's software-intensive organizations are experiencing a paradigm-shift with regards to how to develop software systems. With the increasing availability and access to data and with artificial intelligence (AI) and technologies such as machine learning and deep learning emerging, the traditional requirement driven approach to software development is becoming complemented with other approaches. In addition to having development teams executing on requirements specified by product management, the development of software systems is progressing towards a data driven practice where teams receive an outcome to realize and where design decisions are taken based on continuous collection and analysis of data. On top of this, and due to artificial intelligence components being introduced to more and more software systems, learning algorithms, automatically generated models and data is replacing code and the development process is no longer only a manual effort but instead a combination of human and automated processes. In this paper, and based on multi-case study research in embedded systems and online companies, we see that companies use different approaches to software development but that they often take a requirement driven approach even if they would benefit from one of the other two. Also, we see that picking the wrong approach results in a number of problems such as e.g. inefficiency and waste of development efforts. To help address these problems, we develop a holistic development framework and we provide guidelines on how to improve effectiveness in development. The contribution of this paper is two-fold. First, we identify that there are three distinct approaches to software development; *(1)* Requirement driven development, *(2)* Outcome/data driven development and *(3)* AI driven development and we outline the typical problems that companies experience when using the wrong approach for the wrong purpose. Second, we provide a holistic framework with guidelines for when to use what approach to software development.

Keywords: Requirement driven development, outcome/data driven development, AI driven development, holistic development framework.

1 Introduction

Today's software-intensive business is in the midst of profound changes in relation to development of software systems. With rapid pace, and across industry domains,

sophisticated technologies for data collection and analysis are implemented to provide developers with real-time input on how the systems they develop perform in the field. Also, this data helps developers understand what functionality is used by customers and it allows product managers to confirm whether feature prioritizations were accurate [1], [2], [3], [4]. With automated practices for data collection and analysis, queries can be processed frequently to provide software developers and managers with rapid feedback and as a result, continuous improvements can be made to the systems. This reflects an interesting shift in that traditional requirement driven development practices that have been the de fault approach for decades [5], are being complemented by data driven development practices where teams use data to continuously improve and optimize the system to a certain outcome [4], [6]. As reported in previous research, the challenges with data driven development are numerous [7], but we can already now see that companies that are adept at acquiring, processing and leveraging data become more profitable as decision-making and prioritization based on accurate data from the filed can have a profound impact on annual revenue [4], [8].

Fueled by the increasing availability and access to data, artificial intelligence (AI) and technologies such as e.g. machine learning and deep learning are rapidly adopted in a variety of domains [9]. Although these methods and techniques have been in use for decades, recent years show an increasing use of these in industry with companies such as e.g. Google, Apple and Facebook leading the way but with software-intensive companies in the financial, the medical and the manufacturing domain as fast adopters. For these companies, and for any company with massive amounts of data, deep learning techniques are becoming a necessity and artificial intelligence components are rapidly complementing the traditional software components in a software system.

However, despite the rapid growth of data and the emergence of complementary approaches to software development, most companies have a strong tradition in requirement driven development. In this approach, system requirements are specified in the early stages of development, and although more agile requirements engineering practices are increasingly applied [10], the approach is characterized by a waterfall style of development that works well for systems where requirements are well understood and where revenue is based on delivering a complete product rather than continuous updates of software.

In our research, we see that companies use different approaches to software development but that there are a number of problems associated with selecting the most suitable approach. First, companies with a strong tradition in requirement driven development often take this approach even if they would benefit from an alternative approach. Second, proponents of outcome/data and AI driven development approaches tend to neglect other approaches and instead argue for their approach being the only right one. Third, picking the wrong approach for the wrong purpose results in a number of problems such as e.g. inefficiency and waste of development efforts. In this paper, and based on multi-case study research in companies in the embedded systems and in the online domain, we develop a holistic development framework including three distinct development approaches and we provide guidelines for how to improve effectiveness in development by selecting the optimal one.

The contribution of this paper is two-fold. First, we identify that there are three distinct approaches to software development; *(1)* Requirement driven development, *(2)* Outcome/data driven development and *(3)* AI driven development and we outline the

typical problems that companies experience when using the wrong approach for the wrong purpose. Second, we provide a framework with guidelines for when to use what approach to software development.

The paper is organized as follows. In section 2, we detail the background of our research. In section 3, we describe the research method and the case companies involved in our research. In section 4, we report on the software development approaches in the case companies and we summarize our empirical findings. In section 5, we first identify three distinct development approaches and outline the key problems companies experience when picking the wrong approach for the wrong purpose. Second, we provide guidelines for when to use what approach. In section 6, we conclude the paper.

2 Background

Although the saying that things keep getting faster might sound a little worn-out, the fact is that the software business of today is experiencing bigger, and more rapid, transformations than ever before. The driving force of this is the increasing digitalization of industry that is disrupting companies and society in large to an extent that we have only seen the early beginnings of. As defined by Gartner [11], digitalization is *…" the use of digital technologies to change a business model and provide new revenue and value-producing opportunities; it is the process of moving to a digital business".* To survive this rapid change, companies need new capabilities such as e.g. 'speed' in terms of continuous deployment of software functionality. This allows for continuous collection of customer and product data to use as the basis for determining customer value of new products and services. Moreover, companies need 'data' to allow for artificial intelligence technologies such as e.g. machine learning and deep learning solutions to decrease the time it takes to manually shift through vast amounts of data and to have systems run automatic experiments to help identify, improve and even predict customer value. Finally, access and transparency to data allows for 'empowerment' and autonomy of teams that is critical for any company in order to advance and accelerate team performance and impact [12].

Interestingly, the transformations we see as a result of digitalization have an enormous impact not only on the products and the services that companies produce but also on the ways in which these products and services are produced, i.e. the development approaches themselves. As a result of digitalization and connectivity of products, the traditional requirements engineering process that has been the primary approach for software development for decades is being complemented with other approaches in which continuous use of data, rather than specification of requirements, informs development teams and software systems. As well-known to most software businesses, requirements engineering includes the identification of requirements and the modeling of these in order to develop an agreed upon understanding of what a future software system will look like in order to provide value to the customer and there exist a wide range of techniques to help the development team ensure that the requirements are complete [5]. As recognized in previous research, the goal of the requirements engineering process is to identify what functionality to build before development starts

in order to avoid, or at least reduce the risk, of costly rework [5]. The reasoning is that mistakes that are revealed in the later stages of the development process are more expensive to correct, and that this can be avoided by identifying a stable set of requirements before development resources are allocated and system design and implementation activities start. More recently, agile practices have been adopted to improve flexibility and adaptability of the traditional requirements engineering process and to help software-intensive companies cope with increasing complexity in their software development processes [10].

However, with systems being connected to the Internet and technologies that facilitate data collection and analysis, we see that companies are increasingly complementing their traditional development approaches with other approaches. As one of the most influential trends in software industry, continuous deployment of software is challenging traditional ways-of-working in that it by-passes the notion of early requirements specification. Continuous deployment is a software engineering practice in which incremental software updates and improvements are developed, tested and deployed to the production environment on a continuous basis and in an automated fashion [13]. In this way, customer preferences and needs can be continuously collected, analyzed and deployed and rather than the traditional view of a system being finalized when delivered to customers continuous deployment allows for systems to evolve and improve over time and with delivery to customers as the starting-point for this. In online companies, continuous deployment of software and customer data from A/B tests are the norm for evaluating ideas and understanding customer value and with companies such as e.g. Amazon, eBay, Facebook, Google and Microsoft running thousands of parallel experiments to evaluate and improve their sites at any point in time [4, 13].

The trends described above reflect an interesting shift from a situation where traditional requirements engineering practices inform development of new features, towards a situation in which customer and product data is continuously collected and where companies use this data to inform development during run-time [1], [2], [4]. Also, this leads to interesting opportunities in the field of artificial intelligence as companies today possess such large data sets that manual processing of these become impossible. Today, machine learning and deep learning technologies are emerging as common components in what used to be traditional software systems and the development, production and organizational challenges associated with this shift are far from trivial [9]. Regardless, the software industry is in the midst of a transformation and in order for companies to stay competitive they need to understand, adopt and maximize the benefits from a number of different development approaches. As the systems they develop become connected and will include data collection and processing capabilities, artificial intelligence components and with continuous deployment of functionality as the way to deliver to customers, the approaches they use to develop these systems will advance too. With this in mind, we see the need for guidance on how to complement traditional requirement driven approaches to software development with other approaches as long-term success is seldom achieved by only substituting the former with the later but instead complementing existing expertise with new technology and skills.

3 Research method

The research reported in this paper builds on multi-case study research [14] in software-intensive companies in two industrial domains. The first domain is the *embedded systems domain* and here we studied companies in the context of Software Center (for detailed information please visit https://www.software-center.se/). These companies are large product development companies in e.g. the telecom, the automotive, the security camera, the defense and the manufacturing domains. As a common characteristic, the embedded systems companies are experiencing a challenging transition from being traditional product development companies to delivering products with associated services, and also purely digital services, and where connectivity and data are essential components for innovation and new business models. All companies in the embedded systems domain have significant experience and expertise in relation to requirement driven development as this has been the primary development approach for decades. Typically, the products and systems they develop are highly complex as they involve both hardware and software. In addition, they often have strict rules and regulations to follow as many of their products and systems operate in safety critical environments where standards such as e.g. ISO 26262 defines design, implementation, integration, verification, validation, and release. However, with increasingly connected products and with digital services that generate vast amounts of data, the embedded systems companies are starting to explore other development approaches that help them maximize the benefits associated with data. Although in complex and restricted environments, there are a number of emerging business opportunities and streams of revenue associated with data driven and digitalized services where traditional requirements driven approaches do not capture the potential of rapid feedback cycles and continuous deployment of software.

Over the years, our research collaboration with the Software Center companies has been reported in a large number of publications, e.g. [1], [2], [4], [15], [16] where additional details and careful company descriptions can be found. As reported in these papers, the transition towards digital products and services results in a number of challenges. As one of the most interesting ones, we see that the embedded systems companies seek to complement their traditional and requirement driven development approaches with other approaches in order to reap the benefits of the data they collect. In this paper, and based on our previous research in the Software Center companies, we explore the transition they are in and how the different development approaches they use complement each other.

The second domain is the *online domain* and here we studied companies developing online games, online payment services, media streaming services, travel and accommodation services, online search services and tools for developing artificial neural networks and adaptive systems. These companies are pure Software-as-a-Service companies and with revenue based on license fees, transaction fees and the digital products and services they produce. They continuously add software functionality to their products and they collect and use data as a basis for product development and improvements. In similar to the embedded systems companies, the online companies have access to large amounts of data and they are exploring different development approaches in order to maximize the benefits of this data. In contrast to the embedded systems companies, the online companies have less of a legacy in terms

of requirement driven development. Even if this approach exists also here, they typically use data as the basis for development with teams receiving a quantitative target to realize and are asked to experiment with different solutions to improve a certain metric. In addition, some companies [e.g. …] use artificial intelligence and deep learning technologies as part of development in order to automate tasks and improve speed in problem-solving.

Our research collaboration with the online companies has been reported in a number of publications, e.g. [1], [4], [15], [16] and as reported in these papers, we see that data driven development practices including A/B testing and controlled feature experiments are well-established practices where collection and analysis of data works as the basis for decision-making and feature prioritization. In this paper, and based on our previous research in online companies, we explore the different development approaches they use and how these complement each other. In particular, we recognize how development approaches involving artificial intelligence and technologies such as e.g. machine learning and deep learning are emerging as critical components in many of the software systems they produce.

In total, our research collaborations with the different companies in these two domains cover a time period of more than seven years (2011 – 2018). The collaboration with the embedded systems companies has been an on-going engagement since 2011, and in relation to a number of different topics such as e.g. agile transformation, development feedback cycles, data driven development and value modeling of software features. The specific work on data collection and analysis, and how data can help improve software development, was initiated in 2015 and is on-going. The collaboration with the online companies was initiated in 2015 and is on-going. In all companies, and throughout this period, we have run frequent meetings, interview sessions and workshops involving project managers, product managers, product owners, software developers, software and system architects, data scientists, data analysts and a number of agile team coaches and scrum masters. Meetings are typically scheduled for one hour, workshop sessions for two – three hours and interviews for one hour. The empirical data we build on consists of hundreds of pages of interview transcripts, as many meeting and workshop notes, notes from informal meetings, thousands of e-mails and frequent telephone conversations. Throughout our research, we adopted an interpretive approach to data analysis with the intention to identify recurring elements and concepts in the transcribed interview protocols [17].

In this paper, we build on our previous findings from the embedded systems and from the online companies when exploring the different development approaches they use. In particular, we are interested in exploring how the development approaches they have traditionally been using are being complemented with other approaches.

4 Case study findings

In this section, and based on our previous research in embedded systems and online companies, we summarize our empirical findings in relation to existing and emerging approaches to software development. With selected examples from the two domains, we present the current state as well as the transition that the case companies are

experiencing with regards to how to develop software products and services. In Table 1, we provide a summary where we generalize the characteristics of the development approaches in the two domains. It should be noted that the summary does not reflect details and deviations but rather it captures the dominant characteristics of each domain.

4.1 Software development approaches: The embedded systems domain

The embedded systems companies are in the midst of a challenging transition where the products they develop are rapidly becoming digitalized and where connectivity is key for future innovation and revenue. In this fast-changing environment, the hardware dependencies make development complex as the feedback cycles for hardware are slow while the software cycles are rapid. In most of the companies, the traditional and waterfall approach to development is applied in large parts of the organization while agile practices and methods such as e.g. Scrum are well-established in other parts. It should be noted that many of these companies offer a broad product portfolio which implies that the competence and expertise cover the development and delivery of physical products based on hardware components as well as digital services based on software components. To manage such a disparate product portfolio, the embedded systems companies apply a wide range of development methods and they need to constantly adopt new skills and ways-of-working. Still, and as the most common development approach, requirement driven development characterize both the mind-set and the organizational set-up in these companies. As a common practice, teams receive a requirements specification from product management, and the task for the team is to deliver according to specification. Even if many companies apply agile practices today, they have a long-standing and strong culture where requirements dictate development and where decisions and prioritizations are made based on previous expertise and experience. Typically, qualitative approaches are used to learn about customers with interviews, prototypes and observations being common techniques for data collection. Also, and in line with this culture, requirements are agreed upon in the early stages of development and with sudden changes being a costly disruptor and viewed as something to avoid.

However, and co-existing with the requirement driven culture, the embedded systems companies have been collecting data from their products well before they became connected as many of them are today. For example, the automotive companies started collecting diagnostics data from vehicles already in the early 90's to use as the basis for maintenance whenever a truck or a car was taken to a garage for service. More recently, and as a result of vehicles becoming connected to the Internet and with practices such as continuous deployment in place, car manufacturers can push software updates to the vehicle on a continuous basis without taking the vehicle out of traffic. This allows for preventive maintenance and has become key to prolong the lifetime of a vehicle and avoid costly repairs. Also, effective use of data allows car manufacturers to detect errors while the vehicle is running and before the customer is even aware of them. In similar, telecom companies collect huge amounts of traffic and configuration data as the basis for optimizing performance and operation of their systems as well as for predictive maintenance and monitoring. Based on our research, we see that many of the embedded systems companies are in the process of instrumenting their products to increase and further improve data collection and analysis practices. Also, there are

examples of A/B testing initiatives where companies run experiments with customers to determine whether version A or B of a software feature is the optimal and most appreciated one [18]. In all the companies we studied, the collection and increasing use of data has started to affect the traditional role of product management. With an increasing flow of customer and product data, development teams get a new source from which they learn about the products they develop. In similar, product management get an opportunity to use this data for understanding what adds value to customers. In previous work [2], we report how traditional roles such as e.g. product management change as new roles such as e.g. data scientists emerge. In this research, we see that as companies advance in extracting value from the data they collect, this data will become an effective means for decision-making, as well as work as a basis for product improvements and innovations.

Based on our most recent interactions with the embedded systems companies, we see an emerging interest in artificial intelligence and associated technologies. With connected systems and with large data sets available, new opportunities arise in terms of how to manage, process and utilize this data. For many of the companies, automated practices for collection and analysis of data are already in place as continuous integration and deployment are becoming critical components of their software development approaches. Still, however, supporting infrastructures for increasingly big volumes of data that can handle complexity in terms of variety and velocity [9] are rare and something that would be needed for effective use of solutions such as e.g. deep learning. In our experience, and based on current practices in the case companies, real-time processing of data and artificial intelligence components for supporting this will have a significant impact on future business opportunities as well as for the way in which these companies develop software.

4.2 Software development approaches: The online domain

In contrast to the embedded systems companies, the online companies are less frequent users of requirement driven practices. Although they exist, they don't serve their purpose as the products and systems the online companies develop are inherently different in characteristics and therefore, require other development approaches. Instead, practices such as continuous integration and deployment are fully in place and with products being digital there are no hardware dependencies that slow down the development cycle. This reduces complexity and increases speed and in the majority of the companies, new software functionality is released on a daily or weekly basis. Instead of requirements, the online companies use data collected from their products as the basis for understanding customer needs and preferences. In our experience, most of the online companies have instrumented their products in order to collect relevant data and they have software tools that help them analyze this data. As the basis for data collection, they run A/B tests in which hypotheses on what adds customer value are validated. A/B tests are experiments where two versions of software functionality are compared to determine which one performs the better in relation to predefined criteria such as e.g. conversion rate, click rate or time to perform a certain task [4]. To collect relevant data, users' interaction with the system is instrumented and data on e.g. page views, clicks etc., is collected. In this way, the online companies monitor click-through rates, number of sessions per user, revenue per user and other metrics and use statistical

analysis to determine which variant performs better for a given conversion target [8], [15], [19]. In some of the companies, hundreds of experiments are run in parallel at any point in time and a large number of metrics are used to track product performance and user behaviors. With this data available, the online companies have the opportunity to respond fast and base decisions and prioritizations on data rather than on previous experience and expert opinions in the company. Currently, A/B testing is the dominant technique for optimizing performance, validating new concepts and test new ideas.

Despite the many advantages with using data as the basis for development, the online companies experience challenges with this approach just like the embedded systems companies experience challenges with their requirements driven practices. As reported in our previous research [4], [15], [16] to scale the impact of experiments, to identify and agree on key metrics to optimize for and to find effective mechanisms for evaluating the success of an experiment are difficulties that the online companies face. Also, and as the volume of the data sets increases, there is the need to advance the storage and processing capabilities as well as adopt mechanisms that manage variety and velocity of data.

In our most recent research, we have had the opportunity to learn about some of the emerging trends in these companies and especially about their rapid adoption of artificial intelligence and deep learning solutions. These technologies enable radical improvements in the development cycle by increasing the effectiveness of development and by reducing development time of novel functionality. In one of the case companies [9], deep learning components are developed to provide companies in a variety of domains (e.g. real estate, bookkeeping, weather forecasting etc.) with a platform and with tools for processing, modelling and recognize and predict patterns in large data sets. However, and as recognized in [9], [20], there are no systematic and repeatable methods for creating, evolving and maintaining software systems using these technologies and although successful instantiations exist there are a number of challenges to solve before online companies, as well as embedded systems companies, can fully benefit from artificial intelligence as part of their daily development practices.

Table 1. Generalization of characteristics of the current software development approaches in the two domains.

Characteristic	Embedded Systems Domain	Online Domain
Development cycle	Long (project based)	Short (sprint based)
Requirements cycle	Project/sprint	Sprint/continuous
Quality assurance cycle	Discontinuous	Continuous
Release frequency	Monthly/yearly	Daily/weekly
Decision-making	Expert driven	Data driven
Value creation	Infrequent (product/system)	Frequent (feature/functionality)
Value assessment	Internal validation	External validation

5 Towards a holistic development framework

In this paper, we explore the transition that companies in the embedded systems and in the online domain are experiencing due to digitalization of products and services. Based on our previous research in a large number of companies, we see that companies are complementing their traditional development approaches with other approaches and that this requires a careful understanding of when to use what approach. Below, we identify three distinct development approaches that we see exist in the companies we studied. We summarize the approaches in Table 2. Furthermore, and as an inductively derived model from generalizing our case study findings, we provide a holistic development framework (Figure 1) with guidelines for when to use what approach to software development.

5.1 Three software development approaches

Based on our case study research, we identify that companies use three different approaches to software development. First, they use a *requirement driven development* approach where software is built to specification and where product management is responsible for collecting and specifying requirements as input for the development teams. As can be seen in the empirical examples, this development approach is predominantly used when the new features or new functionality are well understood and defined and where business revenue is not based on frequent releases of new functionality. Especially in the embedded systems companies, there is a long and well-established practice of developing software systems based on requirements. In all these companies, requirements are collected, specified and carefully documented as the main input for development teams and as the mechanism to confirm that a system is developed and delivered according to customer preferences and needs [5], [10]. Over the years, and as experienced in our case companies, a number of limitations have been recognized in relation to the requirement driven development approach with the assumption that customer requirements can be identified before development starts as the most questioned one. Also, techniques and tools for eliciting customer requirements are often insufficient as these tend to focus on what customers say they want rather than what they do in practice which causes a situation in which requirements that can be made explicit are the only ones that can be captured while the more implicit ones remain invisible. Finally, the companies we studied confirm that with techniques such as e.g. brainstorming, interviews, focus groups, observations and prototyping, the amount of data that is collected is relatively small and primarily qualitative in nature which makes it difficult to generalize and identify patterns of behaviors of a large customer group.

However, the requirement driven development approach is well suited for situations in which features and functionality are well-understood and where there is a long-term agreement between the customer and the development organization. Typically, this approach applies for products and services that are intended to last over time and where there is less frequent change imposed on the system. When applied in fast changing environments where customer requirements fluctuate, the requirement driven development approach should be avoided as it falls short on managing frequent iterations and short development cycles.

The second approach companies use is the *outcome/data driven development* approach where development teams receive a quantitative target, i.e. an outcome, to realize and are asked to experiment with different solutions to improve the metric. This development approach is predominantly used for development are new features that are used frequently by customers, and for innovation efforts when there is uncertainty on how to realize a new feature. As can be seen in the empirical examples, this approach is the dominant approach in the online companies where development teams are assigned a certain metric, e.g. conversion rate, and are responsible for improving this metric. To do so, a team typically runs A/B tests with selected customers to identify what version of a website that improves the metric and that moves the needle in the direction set by the business. The decision is based on data that is collected during the experiment and the approach differs from the requirement driven development approach in that continuous collection and analysis of customer and product data informs development rather than requirements specified in the early stages of development. Also, while the requirement driven approach is characterized by smaller amounts of qualitative data as the basis for decision-making, the outcome/data driven approach uses large and quantitative data sets collected at run-time and by instrumenting the code to monitor specific metrics. In our research, we see example of this approach also in the embedded domain where experimentation with different software solutions are becoming increasingly important to determine and validate customer value [1], [2], [21]. The companies we studied report on a number of challenges involved in outcome/data driven development. Often, these relate to the challenge with accumulating and scaling the impact of experiments [4]. In the companies we studied, experiments support smaller improvements of features rather than having an impact on high-level business decisions such as larger re-designs, new product development or innovation initiatives and impact of an experiment is limited. In combination with poor evaluation criteria, the trustworthiness of the experiment might be low. Still, the outcome/data driven development approach is well suited for situations where there is a need to test different hypotheses and where the solution to a problem is unclear. Also, the approach is often applied in innovation efforts as there is the need to test and trial with customers in order to identify the potential value of a new feature, a new product or a new service.

The third approach companies use, and that is rapidly emerging as a new approach to software development, is the *AI driven development* approach where the company has a large data set available and uses artificial intelligence techniques such as machine learning and deep learning to create components that act based on input data and that learn from previous actions. In the case companies we studied, AI is perceived as a very powerful approach with the potential to take on far more complex assignments humans by augmenting our skills, talents and abilities. Examples of this type of development include e.g. object recognition is autonomous cars as well as speech recognition in modern user interfaces. As another example, one of the case companies refers to the use of AI for predicting business sales by pulling data from sales tools together and by using patterns found in this historical and rich data set. Typically, this approach applies for product and service development in which a company has access to a large data set with very many data points and where minimizing prediction errors is critical. Also, and as recognized in the case companies, it is an approach well suited for development situations in which there are too many potential alternatives that manual processing of

these would be either too difficult, too time consuming or too expensive. As the AI approach is fundamentally different from traditional software development in that much of the responsibility for finding a solution to a problem is left to the computer, problems arise when an organization e.g. lack mechanisms and infrastructures for running experiments, has limited resources for large and complex data sets and when the organizational culture, skills and interests do not align with the cross-functional collaboration that is critical for building a production-ready AI system. As recognized in [9], there are additional challenges related to development, production and organization and as this development approach is still in its infancy in many of the companies we studied we foresee significant work in the area of defining systematic and repeatable methods for creating, evolving and maintaining systems using AI techniques.

Table 2. Summary of the current software development approaches that are used in the two domains.

Development approach	Definition
Requirement driven development	Software is built to specification. This development approach is predominantly used when new features or functionality are well understood and defined.
Outcome/data driven development	Development teams receive a quantitative target to realize and are asked to experiment with different solutions to improve the metric. Examples of this development approach are new features (used frequently by customers) and innovation efforts.
AI driven development	A company has a large data set available and use artificial intelligence techniques such as machine learning and deep learning to create components that act based on input data and that learn from previous actions. Examples of this development approach include e.g. object recognition in autonomous cars and speech recognition in modern user interfaces.

5.2 Holistic development framework

As reported above, and due to the increasing access and availability to data, companies are starting to complement their requirement driven development approaches with other approaches. With the adoption of agile development practices [22], [23] the companies we studied have been able to shorten their internal development cycles and, in many cases, integrate development with product operation. In these companies, it is possible to iteratively build new functionality and continuously measure to what extent this functionality is delivering on the expected outcomes. In addition, recent developments in artificial intelligence allow for radical improvements in the development cycle in terms of effectiveness and exploration of novel functionality.

However, and as recognized in this research, the integration of these development approaches is not well understood and there exist little guidance for when to select one approach over another. In addition, and as future systems will include both traditional software components as well as artificial intelligence components, the combination of

approaches will be critical as most software organization will have to manage not only one approach but all three.

To help address this challenge, and to provide companies with a framework on when to select what approach, we present a holistic development framework (Figure 1). Based on a generalization of our case study findings, we outline the three development approaches and we identify the purposes for which each approach is optimal. As can be seen in this framework, the (1) requirement driven development approach is well suited for regulatory features, for competitor parity features and for commodity features, the (2) outcome/data driven development approach is well suited for value hypotheses, development of new "flow" features, i.e. features used frequently by customers and for innovation and the (3) AI driven development approach is well suited when aiming to minimize prediction errors, when there are many data points and when there is a combinatorial explosion of alternatives.

The framework pictures an overall development environment where the system in operation consists of traditional software components as well as AI components, and where continuous deployment practices allow for behavior data to be continuously collected and used as the basis for development. In this environment, and with systems involving different components, the key challenge is to effectively select, combine and deploy different development approaches. Although successful instantiations exist in research and industry, there are no systematic, repeatable methods for creating, evolving and maintaining systems using these techniques.

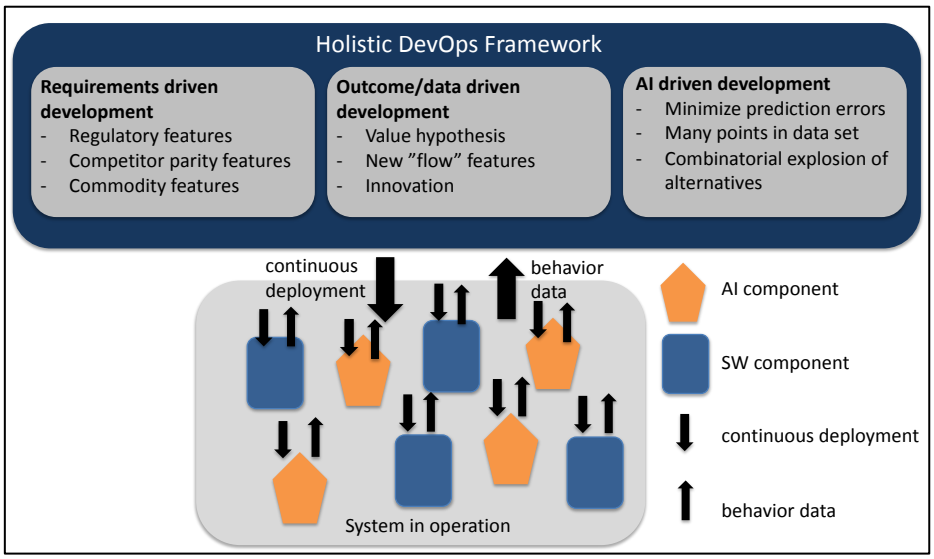

Figure 1. The ‘Holistic DevOps Framework’ including the three approaches to software development.

6 Conclusions

Today’s software industry is in the midst of dramatic transformations with digitalization challenging existing ways-of-working. With increasingly connected and intelligent products, and with availability and access to massive amounts of data, the traditional requirement driven approach to software development is being complemented with other approaches that reflect these new opportunities and technologies. However, in our research we see that companies often take a requirement driven approach even if they would benefit from one of the other two. Also, we see that picking the wrong approach results in a number of problems such as e.g. inefficiency and waste of development efforts.

In this paper, and based on multi-case study research in embedded systems and online companies, we identify three distinct approaches to software development: *(1)* Requirement driven development, *(2)* Outcome/data driven development and *(3)* AI driven development and we provide a framework with guidelines for when to use what approach to help minimize the problems associated with using the wrong approach for the wrong purpose. With this framework, we aim to help companies effectively select, combine and deploy different development approaches in order to manage the digital transformation they are in.

In our future work, we intend to further validate this framework and explore how software-intensive companies in different domains can benefit from complementary development approaches and how successful selection of approaches can become key for competitive advantage.

References

1. Olsson, H.H., and Bosch, J. (2017). Towards Evidence-Based Development: Learnings from Embedded Systems, Online Games and Internet of Things. *IEEE Software,* October 2017, Issue 99.
2. Olsson, H.H., and Bosch, J. From Opinions to Data-Driven Software R&D: A Multi-Case Study on How to Close The 'Open Loop' Problem. *In Proceedings of EUROMICRO, Software Engineering and Advanced Applications (SEAA), August 27-29, Verona, Italy, 2014.*
3. Bosch, J. 2012. Building Products as Innovations Experiment Systems. *In Proceedings of 3rd International Conference on Software Business, June 18-20, Cambridge, Massachusetts.*
4. Fabijan, A., Dmitriev, P., Olsson, H. H., and Bosch J. (2017). The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-driven Organization at Scale. *In Proceedings of the 39th International Conference on Software Engineering (ICSE), May 20 – 28th, Buenos Aires, Argentina.*
5. Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques.* Springer Publishing Company, Incorporated.
6. Bosch, J. (2016) Future Trends in Software Engineering. *IEEE Software.*
7. Dmitriev, P., Frasca, B., Gupta, S., Kohavi, R., and Vaz, G. (2016). Pitfalls of Long-Term Online Controlled Experiments. *In Proceedings of IEEE Conference on Big Data, December 5 – 8, Washington.*
8. Kohavi, R., and Longbotham, R. (2015). Online Controlled Experiments and A/B Tests, *In Encyclopedia of Machine Learning and Data Mining, pp. 1–11.*
9. Arpteg, A, Brinne, B, Crncovic-Friis, L, Bosch (2018). Software Engineering Challenges of Deep Learning, *In Proceedings of the 44th Software Engineering and Advanced Applications Conference (SEAA),* August 29 – 31, Prague, Czech Republic.
10. Paetch, F., Eberlein, A., and Maurer, F. (2003). Requirements Engineering and Agile Software Development. *In Proceedings of the 12th IEEE International Workshop on Enabling Technologies, p. 307-313.*
11. https://www.gartner.com/it-glossary/digitalization (Accessed October 5th, 2018).
12. Bosch, J. (2016). *Speed, Data, and Ecosystems: Excelling in a Software-Driven World.* Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. CRC Press.
13. Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., and Stumm, M. (2016). Continuous deployment at Facebook and OANDA. *In Proceedings of the 38th International Conference on Software Engineering Companion, pp. 21-30.*
14. Maxwell J. A. 2005. *Qualitative Research Design: An interactive approach*, 2nd Ed. Thousands Oaks, CA: SAGE Publications.
15. Fabijan, A., Dmitriev, P., Olsson, H.H and Bosch, J (2018). Effective Online Experiment Analysis at Large Scale. In Proceedings of the 44rd Euromicro Conference on Software Engineering and Advanced Application, SEAA'18, Prague, Czech Republic, 2018.
16. Mattos, I. D, Dmitriev, P., Fabijan, A., Bosch, J and Olsson, H.H (2018). Beyond Ad-Hoc Experiments: A Framework for the Online Controlled Experimentation Process. *In Proceedings of the 19th International Conference on Product-Focused Software Process Improvement, (PROFES). November 28 – 30, Wolfsburg, Germany.*
17. Walsham, G. (1995). Interpretive case studies in IS research: Nature and method, European Journal of Information Systems, vol. 4, pp. 74-81.
18. Bosch, J., and Eklund, U. (2012). Eternal embedded software: Towards innovation experiment systems, In Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change, p. 19-31, Springer; Berlin, Heidelberg.

19. M. Kim, T. Zimmermann, R. DeLine, and A. Begel, (2015). The Emerging Role of Data Scientists on Software Development Teams, No. MSR-TR-2015–30, pp. 1–10
20. Submitted paper…
21. Olsson H. H., and Bosch J. 2013. Towards Data-Driven Product Development: A Multiple Case Study on Post-Deployment Data Usage in Software-Intensive Embedded Systems. *In Proceedings of the Lean Enterprise Software and Systems Conference (LESS), December 1-4, 2013, Galway, Ireland.*
22. Highsmith, J., and Cockburn, A. (2001). Agile Software Development: The business of innovation, Software Management, September, pp. 120-122.
23. Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley.