Challenges of the Digital Transformation in Software Engineering

Michael Gebhart
iteratec GmbH
Stuttgart, Germany
e-mail: michael.gebhart@iteratec.de

Pascal Giessler, Sebastian Abeck
Cooperation & Management
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
e-mail: pascal.giessler@kit.edu, abeck@kit.edu

***Abstract*—Digital transformation describes the changes that the increased digitization within society causes or influences in all aspects of human life. The digital business transformation can be understood as the impact of the increased digitization on the business domain. Companies are challenged to transform, i.e., to create new business models that consider and leverage the increased digitization. As a result, from a software engineering perspective, the digital transformation changes the way how software is developed. Current trends are the development of applications for mobile devices and Internet of Things (IoT) applications. However, with these new application fields, new challenges for software engineering arise that have to be met to successfully conduct software projects in a digitized world. It is necessary for software companies to solve these challenges if they want to be successful on the market. In this article, these challenges are worked out. Furthermore, solution approaches, such as Application Programming Interface (API) strategy and an appropriate team culture, are derived to help software developers and companies to prepare for future software projects and to remain competitive.**

Keywords-digital transformation; digitization; software engineering; challenge

I. INTRODUCTION

Digital transformation can be understood as the changes that the digital technology causes or influences in all aspects of human life [1]. From a business perspective, companies are required to react on the increased digitization and to adapt accordingly. The resulting digital business transformation represents the idea of creating new or adapting existing business models based on the increased digitization within society, such as the usage of mobile devices, social media, and Internet of Things (IoT) [2].

As software systems are meant for supporting human life, the digitization influences the way how software should be developed. For example, more and more software projects are conducted that focus on the development of applications for mobile devices, such as smart phones and tablets, or the IoT. According to current studies, such as the one conducted by EC SPRIDE in collaboration with the University of Darmstadt and Fraunhofer Institute for Secure Information Technology (SIT) [3], this trend will continue and apps and rather small special purpose software predominantly in use on smart phones and tablets will further propagate.

These new application fields of software systems bear new challenges for the software engineering discipline. For example, in the last years, several new technologies and frameworks have evolved that are especially meant for the upcoming new devices and technologies. Instead of developing the entire functionality from scratch, more and more web services are available that are integrated into the developed solution [3]. Furthermore, software systems become increasingly omnipresent [1], which requires software developers to consider the consumer of software. The attractiveness of the software for the consumer, i.e., its user experience, has become one of the most important success factors for software systems.

To be successful on the market, software companies have to solve these challenges. However, in most cases, companies try to respond on the digitization with changes of the development methodology. For example, agile development is wide-spread today. But, the increased digitization requires more than focusing on the development methodology. It is more about the attitude when developing software and the necessary mindset and environment. It is about the roles and their responsibilities during the development. The digitization requires to deal with the upcoming changes, such as the increasing technology heterogeneity, instead of trying to avoid them.

This article examines the challenges caused by the digitization in society. The challenges are systematically worked out and described. Purpose of this article is to support software developers and companies to prepare for these challenges the digitization and the resulting digital transformation bring with them. For that reason, in addition, after introducing the challenges, possible solution approaches are presented in this article. The challenges and solutions are not limited to technological aspects. Instead, they focus both technological and organizational respectively human-oriented aspects. For that reason, this paper is meant for software developers and business managers who want to prepare themselves or their company for the future.

The article is organized as follows: Section II examines existing work in the context of digital transformation and its impact on software engineering. The resulting challenges of the digital transformation on software engineering are summarized in Section III. Section IV introduces solution approaches to overcome the challenges managed in Section III. Section V concludes this article and introduces future research work.

 ISBN: 978-1-61208-498-5

II. BACKGROUND

This section analyzes existing work in the context of digital transformation and possible impacts on software engineering. Based on this work, in the following section, the future challenges for software engineering and solution approaches are derived and summarized.

In [1], Stolterman and Fors introduce the term digital transformation as the changes that the digital technology causes or influences in all aspects of human life. They state that the most crucial challenge for Information System Research is the study of the overall effects of the ongoing digital transformation of society. This is, why we focus on the effects of digital transformation on software engineering from a holistic view. We do not focus on software development methodologies like Scrum. Instead, we describe challenges on software engineering as a whole with the involved people in mind.

Opportunities to create new business models based on the results of the increased digitization in society are described by Berman in [2]. Even though Berman focuses on the business perspective and does not describe concrete recommendations for the software engineering, the work gives some important hints about how the business will evolve. This allows to derive the impact on software engineering. For example, according to Berman, the focus should be on digital products and services with better customer experience. This shows the importance of considering the customer experience as developer as part of the software engineering.

In [3], Ochs presents a study about emerging trends in software development. The study is conducted by EC SPRIDE in collaboration with the University of Darmstadt and Fraunhofer Institute for Secure Information Technology (SIT) sponsored by the German Federal Ministry of Education and Research. The study shows essential characteristics for future software systems. We reuse these software characteristics, adapt them to software engineering with focus on digital business transformation and combine them with challenges mentioned in other work.

A list of challenges and methodologies to master the digital transformation is introduced by Hanna in [4]. The aspects described by Hanna are business-driven and consider companies as a whole. For example, Hanna describes how to develop the human resources, leadership and institutions, policies and regulations. Even though it is not focusing Information Technology (IT), we use this work to derive challenges that relate to software engineering.

The importance of solutions being consumer-oriented is emphasized by Leimeister et al. [5]. They describe that “in digital societies, companies must understand that the digital customers and their preferences are key and at the center stage for developing innovative solutions”. This work shows that it is necessary to rethink about the way solutions are developed today.

The analysis of existing work shows that the digitization within society influences the software engineering. With the digital transformation, new challenges arise that have to be considered by software companies. Some work also describes certain challenges and trends for software engineering that have to be solved. However, a list of challenges of the digital transformation in software engineering with concrete advices how to solve these challenges is missing. This is the motivation to investigate this aspect in more detail.

III. CHALLENGES OF THE DIGITAL TRANSFORMATION IN SOFTWARE ENGINEERING

In this section, the challenges of the digital transformation in software engineering described. The challenges are derived from existing work and experiences. The challenges consider both technical and organizational, i.e., human-oriented aspects as both are important for successful software projects. They constitute the basis for the solution approaches introduced in Section IV.

A. Reduced Time-To-Market

More than ever, new innovations are expected by the customers to be available as soon as possible [4]. Especially due to the increasing digitization in society and thus, due to the focus on private customers, the time to market has to be further reduced. This applies to completely new products and new features for existing products.

As a result, software developers and companies have to prepare their development process in way that allows to deploy new functionality continuously.

B. Flexibility and Agility

The reduced time to market requires software systems to be flexible enough to consider new requirements afterwards. Furthermore, the high competitive pressure requires to replace originally planned features by other ones. Thus, if not already done, the development process has to be agile to react on changing requirements.

Wherever useful, software developers and companies have to avoid waterfall development models and work in an agile way to be flexible enough to consider new and just needed functionality.

C. New Disciplines

In the last few years, a high number of new business words have been established with partially new concepts. Terms like Big Data, Internet of Things (IoT), Industry 4.0, and Machine Learning comprise complex topics that might be understood to create competitive products.

Thus, software developers and companies have to understand these new terms and concepts. As described by Hanna in [4], technological change creates new demand for learning. This requires enough time and freedom and efficient trainings for developers.

D. New Devices and Technologies

With the increased digitization, new devices are used by the customers [4]: mobile devices, virtual reality glasses, smart watches, intelligent in-ear headphones, and all the intelligent devices in a smart home [3]. They all create a completely new device environment with partially completely new technologies. The number of programming languages, partially device-specific frameworks increases.

 ISBN: 978-1-61208-498-5

Thus, software developers and companies have to prepare for completely new devices and technologies. This means that companies have to establish trainings and give developers the necessary freedom that allows them to learn new concepts. The training of developers is an important success factor as the new devices and technologies bring new possibilities with them that are necessary to stay competitive in the market.

E. High Degree of Technology Heterogeneity

The high number of devices and technologies results in another challenge: The rapid time to market does not allow to wait for a homogenization of the technology portfolio. As a result, software developers and companies are challenged to deal with a high number of devices and technologies in parallel. Within one project, several different technologies might be used with all their specifics.

For that reason, developers and companies have to find a solution to deal with this high degree of heterogeneity. Possible solutions are on the one side to standardize the interfaces between these technologies and on the other side to enable a continuous integration that considers different technologies.

F. Stronger Networking

Software systems developed today are mostly connected to other already existing software systems. This kind of networking can be the classical usage of programming interfaces to exchange data (e.g., by means of web services [6]) or the integration with social networks for authentication. In every case, software systems are less often isolated components. Instead, customers want them to interact with a bunch of existing components [2][3].

For that reason, for developers, it is necessary to know the requirements customers have on the networking aspect of software.

G. Extended Service Market

Today, we find more and more services providers that offer infrastructure, platform, and software as a service [2]. Compared to traditional software development, the trend is to reuse existing functionality instead of developing it from scratch [3]. Even though, using software from within the cloud includes operational costs, the development speed and the quality of the results can be much higher when reusing provided services.

Thus, the way how to develop software systems changes from coding the required functionality to assembling existing solutions. Software developers and companies have to understand the opportunities this way brings with it. As a result, the new service market has to be overseen. Software developers and companies have to inform about services that are currently available on the market.

H. Consumer-/ Customer-Orientation

Today, more and more software systems are used by ordinary people. Especially with the advent of smart phones and tablets, there is a new understanding about how software systems are expected to be used. The opportunity to enhance products and services for a better customer experience is mentioned by Berman in [2]. Also, Leimeister et al. emphasize the necessity to create solutions that are consumer-oriented [5].

This means that software developers and companies have to focus more than ever on the user of the system. An appealing user interface with a clear user experience has become one central success factor for consumer applications. Thus, developers and companies have to consider this aspect in their development process and to increase their competence in this area if not already done.

I. Business-Awareness

One central idea of the digital business transformation is to consider and leverage the increased digitization within society to create new business models. According to Hanna [4], product innovation has become crucial for sustained growth, competitiveness, and moving up the value ladder. As innovations require technological knowledge, the creation of business models is not a pure business task any longer. More than ever, developers themselves are invited to communicate new business ideas and models to the management.

For that reason, the developers within a company are required to think more business-oriented and to think outside the box. On the other side, the management is required to support developers in this task. This requires on the one hand freedom to give new ideas a trial and on the other side to open up to new ideas from developers.

J. Combination with Legacy IT

The increased speed with which new devices and technologies arise results in a faster aging of existing IT [3]. Especially in grown landscapes as they exist in bigger companies, the existing IT cannot be replaced rapidly. For that reason, developers and companies will be more often challenged to combine their new and modern software systems with IT that can be considered as legacy in the meanwhile. In [4], Hanna states that “adjusting the social and institutional environment to take advantage of a technological revolution and its associated techno-economic paradigm involves painful adjustments, often disruptions to, and even destruction of, legacy systems, institutions, practices, and processes”. For example, in some cases, the new intelligent solutions have to exchange data with existing systems. In other cases, the new solution has to be deployable in an infrastructure that is not as modern as the new software system.

Thus, software developers and companies have to keep in mind where the software system is expected to be executed. Possibly, it is necessary to work out a more complex integration and deployment process to combine the new and modern software system with existing legacy IT.

IV. Solution Approaches

In Section III, challenges of the digital transformation in software engineering were introduced. To overcome these challenges, next, solution approaches are described. These approaches are meant to help software developers and companies to prepare for future software projects and to remain competitive.

 ISBN: 978-1-61208-498-5

A. Microservices

Microservices can be seen as "small and autonomous services that work together" [8] via well-defined web interfaces. The size of a microservice is usually given by bounded contexts derived from business boundaries [8][9]. Therefore, the business has direct influence on the resulting system design which is also known as Conway's Law. But, there is also an influencing factor towards the opposite direction especially when evolving the service landscape [8].

The concept behind microservices results in a large number of benefits that address the mentioned challenges in Section III: First, reduced time to market when developing new functionalities through service autonomy. Each service can be developed and deployed separately without a central release coordination between different development teams. If this is not possible, the bounded context of the microservice should be usually revised and adapted to fit current needs.

Second, the support of technology heterogeneity when designing service landscapes especially microservice landscapes. We can decide to use different technologies inside each microservice based on our needs instead of using a company-wide standardized one [8]. For example, one service can be written in Java, another one in C#. It is not necessary to have company-wide unification. The communication between microservices with different technological setup is ensured by the service interface. This service interface must follow a technology-independent approach, such as the architectural style Representational State Transfer (REST) introduced by Fielding [7].

Third, the reuse aspect that is the core idea behind service landscapes [10]. Each service can expose different functionality via a well-defined service interface so that other services can benefit from that offer. For discovering existing services in a service landscape, a service discovery solution, such as Consul or Eureka, is usually set up.

B. API Design and Strategy

An Application Programming Interface (API) can be seen as a contract prescribing how to interact with the underlying system. In the context of a service landscape, the system can be an operable service that exposes business functionality via an API.

Today, APIs are a big deal and the growth of public APIs regarding their activity measured in requests is still unbroken [11]. Popular examples for public APIs are Facebook with the Graph API or Google with several APIs, such as YouTube Data API or Cloud Vision for image recognition.

There are several reasons for serving an API, such as offering business functionality for a mobile application, more flexibility in providing content, and allowing external developers or partners to transform new use cases in reality [11].

When designing an API, it is very important to have clear vision and business objective. In addition to the business view, it also crucial to comply with some design principles and best practices to simplify the usage of the API. For instance, in [13], several best practices for RESTful Web APIs are identified, collected, and explained.

C. Automation

Automation can be a key factor in the success of a company since it can drastically reduce the time and effort incurred by recurring tasks. This in turn increases the team productivity because the team does not have to deal with these tasks anymore.

For instance, deploying new applications or services should be as easy and fast as possible to reduce the time-to-market. At best, there is only one command for the deployment. That is why, Platform as a Services (PaaS) solutions, such as Heroku, DigitalOcean, or Amazon Web Services (AWS), are particularly popular among companies since it simplifies and reduces the necessary effort for deployment or infrastructure configuration.

But, deployment is just one of many examples, such as application monitoring or log analysis. It should be clear that an initial invest has to be taken, but it will pay off.

D. Technical Governance

The technical governance provides corporate-wide guidelines that have to be respected by all development teams to keep the heterogeneity within limits. Although, it is desirable when development teams take individual design or technological decisions, it makes, for example, no sense to choose a different coding style or to use a distinct deployment workflow. Sometimes, some technological decisions taken by team are not appropriate within the company due to bad experience in the past or other business related reasons.

Particularly important are corporate-wide guidelines when composing software components or services since the guidelines lay the foundation of the API that forms a contract for communication (see Section IV.B). For instance, the usage of different naming styles of exposed functionality can impede the reusability. Or in the worst case, when choosing a technology-dependent API, it can even prevent the reuse entirely.

In our opinion, any decision that affects the whole system should be covered by one or more guidelines and governed by a technical committee.

E. Accepting and Dealing with Heterogeneity

As mentioned, the technical governance tries to limit the heterogeneity in case of, for example, API design, but it does not want to avoid it when it comes to technological decisions. The company has to get used to the fact that several different technologies are applied. This is not a weakness but a strength because the new technologies offer new possibilities. Furthermore, a heterogeneous technology landscape avoids that too many people get used to a technology that becomes outdated one day. Developers are challenged to learn new technologies. As a result, developers remain attractive for the labor market. Finally, also, the company remains attractive for developers and young applicants that prefer to work and learn new and modern technologies.

 ISBN: 978-1-61208-498-5

Thus, heterogeneity is nothing that should be avoided. Instead, the company should learn to deal with and get used to it. The attempt to homogenize the IT at any price will result in a reduced adoption of new technologies, thus less innovative software projects and business models. It is more important to accept the coming changes and to make it part of the company philosophy.

F. Critical Analysis of new Trends

Every day, new technical solutions in form of libraries, frameworks, and tools appear on the internet. Some of them offer a comparable solution to a recurring problem or use case. Then again, others offer a completely new approach to solve the same problem.

The rapid growth of new technologies, the necessary learning and training effort as well as the lack of detailed and objective comparison between technologies in the same domain make it difficult to choose one technology over the other. In many cases, the taken decision has also to be reasoned especially when some people or the whole company must be convinced to rely on this technology for upcoming projects. There is no silver-bullet on how to choose the right technology since there are several factors that have to be considered. Furthermore, it is often an illusion to choose one technology for the next ten or more years since at some point a new technology or new approach will arise and form a community around it. Then, the cycle starts from the beginning.

We recommend to be open for new technologies or approaches and not hang on prior decisions based on the motto "It still works and will also be a suitable solution for the next years. We do not need a new technology". In our opinion, new technology should be analyzed as soon as possible to get a clear picture if it makes sense to follow the further development and how this approach can improve the already used technology in a specific application field. Thereby, it is important to look behind the scenes and get an idea of how the technology works and how it is build up since this is crucial for a proper and reasoned decision.

G. Design

A lot of great services and (digital) products pop up every day on the market. The design is the first impression that a potential customer will get from a product. It is often a misunderstanding that the design is only the user interface of an application. Instead, we should see the design as a method for problem solving that can be applied on different contexts, such as the user interface of an application but also on an API of a service. For instance, if the API design is not easy to understand or easy to try and use, it can have a negative impact on the adoption rate. Similar to this, if the frontend of an application looks confusing, customers could look after the opponent's solution. The design is therefore an essential ingredient of a product that decides if a potential customer is interested or not.

When developing new products, the whole development team should concentrate on a good design for the target user group instead of only focusing on the functionality aspect of the product.

H. Team Culture

The team culture of the development team can have a positive impact on the productivity and the self-awareness of each member. It can be difficult to form a good team culture since there are several factors that have to be considered, such as the identification of each member with the company or how proud someone is about shipped or developed products. Nevertheless, there are some key points that should be achieved within a company.

First, you should give each member of the development team the feeling that (s)he is unreplaceable and essential for the success for the product.

Second, each manager should be open for discussions and ideas since innovation comes from ideas. Each developer should be encouraged to be more business-oriented and think outside the box.

This leads to the third important key point: The employees are the most important asset of a company and should be treated accordingly. For development teams, you should give them the opportunity to learn and teach as much as possible to keep up with the industry leaders since no developer wants to work with "old" technologies for the rest of his working life. If this is not possible in current projects, then the company should find a compensation for the developers, perhaps by starting an open source project that allows to study new technologies. Thus, the management is responsible to give the necessary freedom and to provide trainings wherever possible.

V. Conclusion and Outlook

Digital business transformation can be understood as the idea of creating new or adapting existing business models based on the increasing digitization within society. As these new business models influence the way software systems have to be developed, software developers and companies have to adapt to these changes. For that reason, in this article, we derived and described challenges for software engineering the digital business transformation brings with it. Furthermore, we listed solution approaches for software developers and companies for overcoming these challenges.

Our list of challenges and solution approaches is expected to help software developers and business managers of software companies to remain competitive in times of digital transformation. Furthermore, more than ever, it is important to remain attractive for the labor market. Competitiveness requires companies to be attractive for talents. For that reason, the challenges and solution approaches are not limited to technological aspects. Instead, we considered aspects that are both technological and organizational, i.e., human-oriented.

For the future, we plan to refine the solution approaches. In the past we started with methodologies for designing services in service-oriented architectures [12]. We extended this work for API design and strategy including best practices for RESTful web services [13] and approaches to measure the compliance regarding these best practices [14]. We will continue this technological work to provide guidelines for designing a clear and maintainable API. In addition, we will focus on the non-technological, the human-oriented aspects. We will investigate solution approaches for being attractive

 ISBN: 978-1-61208-498-5

for talents as this is significant for remaining competitive on the market. Furthermore, we will investigate how to adjust the mindset within companies so that new innovative and competitive business models can be created. This will help companies to benefit from the increased digitization and to prepare for the future.

REFERENCES

[1] E. Stolterman and A. C. Fors, "Information technology and the good life," in Information Systems Research: Relevant Theory and Informed Practice, pp. 687-692, 2004.

[2] S. J. Berman, "Digital transformation: opportunities to create new business models," Strateg. Leadersh., vol. 40, no. 2, pp. 16–24, 2012.

[3] C. Ochs, "Emerging trends in software development & implications for IT security: an explorative study", EC SPRIDE, 06/2014.

[4] N. K. Hanna, Mastering Digital Transformation: Towards a Smarter Society, Economy, City and Nation. Emerald Group Publishing Limited, 2015.

[5] J. M. Leimeister, H. Österle, and S. Alter, "Digital services for consumers," Electron. Mark., vol. 24, no. 4, pp. 255-258, 2014.

[6] S. Varghese, "Web Development with Go: Building Scalable Web Apps and RESTful Services," Berkeley, CA: Apress, pp. 159–209, 2015.

[7] R. Fielding, "Architectural styles and the design of network-based software architectures," University of California, Irvine, 2000.

[8] S. Newman, "Building Microservices – Designing fine-grained systems," O'Reilly, 2015, ISBN 9781491950357.

[9] E. Evans, "Domain-Driven Design: Tacking Complexity In the Heart of Software," Addison-Wesley Longman Publishing Co., Inc., 2003, ISBN 0321125215.

[10] T. Erl, SOA – Design Patterns, Prentice Hall, 2008. ISBN 978-0-13-613516-6.

[11] D. Jacobsen, G. Brail, and D. Woods, "APIs – A Strategy Guide," O'Reilly, 2012, ISBN 9781449308926.

[12] M. Gebhart and S. Abeck, "Metrics for evaluating service designs based on soaml," International Journal on Advances in Software, 4(1&2), pp. 61-75, 2011.

[13] P. Giessler, M. Gebhart, D. Sarancin, R. Steinegger, and S. Abeck, "Best Practices for the Design of RESTful web Services," International Conferences of Software Advances (ICSEA), pp. 392-397, 2015.

[14] M. Gebhart, "Query-based static analysis of web services in service-oriented architectures," International Journal on Advances in Software, 7(1&2), pp. 136-147, 2014.

 ISBN: 978-1-61208-498-5