Corinna Rutschi and Jens Dibbern

6 Transforming and recombining routines to scale the implementation of software robots

Abstract: With the increasing potential to automate business processes using software robots, companies face the challenge of scaling the implementation of such robotic systems in order to enable their efficient evolution. The implementation of software robots is based on the often time consuming work carried out by the project team, which often leads to higher than expected costs and time delays. This can be made more efficient by scaling the extension of the robot's functionalities. However, scaling can only take place once one has understood what can be scaled, how it can be scaled, and to what extent. Routine theoretical concepts help us better understand the extent to which processes previously carried out by humans can be transformed and transferred to robots. We build on literature on routine dynamics as well as digital scaling to understand the mechanisms required to scale the implementation of software robots. Therefore, based on an empirically illustrated theoretical conceptualization of scaling the software robot implementation, we elaborate in this chapter how routines evolve and dynamically influence each other in order to explain how scaling can be approached when implementing software robots. In doing so, we rely on data from two case studies. In one case study a chatbot was contextually expanded over time. In the second case study a series of robotic process automation (RPA) robots were implemented.

Keywords: Software robots, robotic process automation, chatbot, software implementation, digital scaling, routine theory

6.1 Introduction

Companies are increasingly introducing software robots to automate some of their business processes. They often start with implementing software robots in a specific area or department, such as customer support (Willcocks and Lacity, 2016). Based on such initial experiences, they may identify additional departments or contexts in which they could implement additional robots. However, if they start from scratch for each new robot to be programmed this results in high costs and time expenditure. Accordingly, companies may ask themselves how they could speed up the robot implementation process. Ideally, robots are implemented with the intention of extending their scope and reach, which simultaneously poses the challenge of scaling their implementation. While the initial development and programming of robots often mirrors an innovative process, which requires exploration and experimentation, their further development can be made more efficient by drawing from scaling mechanisms. Such

https://doi.org/10.1515/9783110676693-006

scaling should allow for a more efficient implementation of software robots since it allows for extending functionality of current robots or developing additional robots in novel contexts, without significant additional costs. As a result, the conditions for scaling the implementation of software robots are an important area of inquiry.

Prior research on software robot implementations has mainly focused on the above-mentioned one-time implementation process. Thereby, drawing on routine theory, the implementation of software robots has been viewed as the transfer of routines from humans to robots (D'Adderio, 2011; Rutschi and Dibbern, 2019, 2020). This has led to initial insights into the steps to be taken and the underlying conditions for automating particular (business) processes (Rutschi and Dibbern, 2019, 2020). It was shown that the basis of automation requires an understanding of the structure of both, existing business processes to be automated and the operating principle of a robot, i. e., one needs to think like a robot.

In this study, we seek to develop a better understanding of the robot implementation process to consider how it changes when companies already implemented software robots that can be taken as a basis to develop additional robots. Thereby, we seek to close the gap in understanding the dynamic evolution of software robots by developing some foundational knowledge on the scaling of the implementation of software robots. The notion of scaling has recently been gaining increased interest in the IS literature, when it comes to understanding the exponential growth of digital startups and entrepreneurs. Such digital growth often rests on the creation and recombination of digital resources, which has also been referred to as the scaling of digital infrastructures (Henfridsson and Bygstad, 2013). In a like vein, an existing software robot may also be viewed as a digital resource that serves as the basis for scaling the process of software robot implementations. However, little is known about what this process of scaling actually looks like in the context of software robots. Accordingly, we aim at investigating how the implementation of software robots can be accelerated. In order to gain insights into how such acceleration or scaling can be achieved, we need to understand what can be scaled (i. e., scaling resources), how it can be scaled (i. e., scaling mechanisms), and to what extent it can be scaled (conditions for contextual growth). In line with existing research on software robot implementations (Rutschi and Dibbern, 2019, 2020), we draw on routine theory (D'Adderio, 2011; Leonardi, 2011) as a basis for understanding how human-executed business processes can be transformed into robot-automated processes. Specifically, we draw on recent advances in routine theory that explain how the performance of routines to achieve particular ends leads to the generation of means that can be reused to achieve new ends (Dittrich and Seidl, 2018; Howard-Grenville, 2005). By reusing generated means, the overall pace of achieving a specific end can be accelerated. A similar dynamic can be observed in the implementation of software robots, where certain components can be created that can subsequently be reused, which may result in scaling the implementation process. Such digital scaling may be described as a dynamic reinforcing process by which the reach of a software robot is extended either through expanding its functionalities or

by transferring its functionalities to additional software robots that may reuse part of its components (Henfridsson and Bygstad, 2013). In order to better understand digital scaling in the context of software robot implementations, we formulate the following research question: How can the implementation of software robots be accelerated and thus the transfer of routines to software robots be scaled?

We seek to address this research question by taking a dynamic perspective and by exploring the generative mechanisms that help in scaling the transfer of routines to robots, i. e., the robot implementation process. Our overarching objective is to explain what digital scaling means in relation to transferring routines to robots. Routine theoretical concepts help us better understand the extent to which processes previously carried out by humans can be transformed and transferred to robots not only once, but concurrently. We build on literature on routine dynamics as well as digital scaling to understand the conditions for scaling the implementation of software robots. Therefore, based on an empirically illustrated theoretical conceptualization of scaling the software robot implementation, we elaborate how routines evolve and dynamically influence each other in order to explain how scaling can be approached when implementing software robots. In doing so, we rely on data from two case studies that we use for an illustrative purpose in this chapter. In one case study a chatbot was implemented. In the second case study RPA robots were implemented.

6.2 Scaling as a process of routine emergence

For a successful evolution of software robots, the process of enlarging its reach and functionalities over time must be understood. In an information systems (IS) perspective, scaling means extending an IS in size and/or scope within the same or a new environment. In relation to software robots, an environment could describe the setting (or context) in which a software robot acts involving all surrounding actors. Thus, scaling describes practices that allow a technology to be "spread, enhanced, scoped, and enlarged" (Sahay and Walsham, 2006, p. 43). In contrast, the term scale refers to the size and scope of an IS that can be achieved by scaling (Sahay and Walsham, 2006). Up to now, scaling has mainly been used to achieve economies of scale through standardization (Chandler, 1990). Scaling may thereby lead to different outcomes, such as an increased user base (Huang et al., 2017) or a successfully evolved digital infrastructure (Henfridsson and Bygstad, 2013). The implementation of software robots essentially describes the transformation of human-executed processes to such carried out by a robot (D'Adderio, 2011; Rutschi and Dibbern, 2019, 2020). Such processes could be associated with routines. A routine can thereby be described as a series of interdependent actions performed on a pattern basis whereby various actors can be involved (Feldman and Pentland, 2003; Feldman et al., 2016). Routines are composed of ostensive and performative aspects. The ostensive aspect refers to formal rules and

procedures that can be described as the "guidelines" of how to perform the routine. The performative aspect refers to the actual performance of these rules and procedures. Both the performative and ostensive aspects of a routine influence each other if the routine is performed by a human (D'Adderio, 2011; Feldman et al., 2016; Pentland and Feldman, 2005). To illustrate this using an example, consider an employee A who must prepare a monthly report. Since employee A must prepare the report in the same way every month following certain guidelines, we can characterize this process as a routine. The guidelines describe the ostensive aspect of the routine. The actual preparation of the report describes the performative aspect of the routine. Each time employee A prepares the report, he or she could potentially identify means to prepare the report in a better way or more efficiently next time. This would cause the performative aspect of the routine (i. e., the way of preparing the report) to change. Over time, employee A could thus deviate from the original guidelines. This would also cause the ostensive aspect to change. Thus, employee A has an influence on both the ostensive and the performative aspects of the routine of preparing the monthly report. Suppose employee A hands over the task of preparing the report to another employee B. Employee A would have to explain to employee B how to prepare the report (i. e., ostensive aspect) before employee B could actually prepare the report (i. e., performative aspect). Eventually, employee B will bring in his or her own way of doing things and thus change the performative aspect of the routine, which may lead to a change in the ostensive aspect as well.

Humans do not always perform routines in the same way. They can change routines due to changing contexts or circumstances (Dittrich and Seidl, 2018; Howard-Grenville, 2005). Thus, humans can influence and change the performative as well as the ostensive aspect of a routine. Routine theory helps to unlock the black box of how humans perform routines (Leonardi, 2011). Before routines can be transferred to robots at all, an understanding of how the routine is composed and the extent to which it has previously been performed by humans must first be gained. An understanding must be gained of both the ostensive and performative aspects of the routine as it is performed by a human.

Once such an understanding has been established, one must comprehend the dynamics of software robots and the extent to which robots are capable of performing routines. One must think like a robot to understand how a routine previously performed by a human can be adequately translated so that the robot can later understand and perform it (D'Adderio, 2011; Rutschi and Dibbern, 2019, 2020). Our focus here lays on software robots that execute rule-based processes and are not able to learn autonomously. Such robots are developed by humans (for example the developers) by means of programming within a given software environment. The software robot then determines how a certain routine must look for the robot to perform it. Thus, the robot can initially influence the ostensive aspect of the routine. Once implemented, however, the robot cannot influence the routine itself anymore but executes

it according to the implemented guidelines (i. e., the ostensive aspect). Unlike humans, software robots execute routines by strictly following the given guidelines. Consequently, when a robot executes a routine, no reciprocity between the ostensive and the performative aspect can be observed, rather the ostensive aspect corresponds with the performative aspect.

Rutschi and Dibbern (2020) explain the phenomenon of automating processes by means of software robots and introduce an iterative framework of routine automation, which they use to explain to what extent routines can be transferred from humans to robots. They explored the extent to which an individual process or routine can be transferred from a human to a robot (Rutschi and Dibbern, 2020). In case a company wants to automate numerous processes and thus transfer them to robots, it can lead to enormous costs and time expenditures if the company must approach the automation of each process or the programming of each robot individually. This could be made more efficient if one could draw on prior automation approaches and thus accelerate the whole automation process. In order to ensure a successful evolution of software robots, it is essential to understand what scaling means in this context. This requires an understanding of what is scalable, i. e., of what can be scaled, how it can be scaled, and to what extent it can be scaled (Sahay and Walsham, 2006).

6.2.1 What to scale

Robots are designed to perform certain tasks by following certain behavior patterns or rules. In addition, robots include features such as "adaptivity, robustness, versatility and agility" (Pfeifer et al., 2007, p. 1088). Generally, a robotic system or a software robot can be any machine replacing work performed by humans (Willcocks and Lacity, 2016) while gathering information and following instructions to execute tasks (Tirgul and Naik, 2016). Examples for robotic systems are robotic process automation (RPA) (Willcocks and Lacity, 2016) and chatbots (Sengupta and Lakshman, 2017). Implementing RPA robots allows companies to automate back office business processes (Slaby, 2012; Willcocks and Lacity, 2016).

RPA robots execute processes like humans while interacting with IT systems through their user interface (Asatiani and Penttinen, 2016; Willcocks and Lacity, 2016). In doing so, RPA robots log in (and out) of systems like humans do (Willcocks and Lacity, 2016).

Implementing a chatbot allows companies to automate conversational business processes (Heller et al., 2005; Shawar and Atwell, 2007). After releasing a chatbot into the live system, the human user can interact with it via a user interface, such as a pop-up window integrated on a web site (Sengupta and Lakshman, 2017), Facebook Messenger, Skype, or Slack (Patil et al., 2017).

By introducing software robots, companies can benefit from improved performance in terms of quality and efficiency, as robots outperform humans in executing

certain tasks and processes (Fung, 2014; Guzman and Pathania, 2016; Sengupta and Lakshman, 2017; Sharma et al., 2016; Slaby, 2012). Scaling in the sense of implementing robots might be described as engineering robots "capable of performing a large variety of tasks" (Pfeifer et al., 2007, p. 1091). However, scaling does not refer to the extent to which a system can be configured, customized, parameterized (Sahay and Walsham, 2006), or adapted. Adaptation may be necessary in the case of environmental changes so that a system can perform processes exactly as it did before the change. However, this does not mean that its functionalities are extended and therefore cannot be called scaling. What can be described as scaling is the step-by-step process in which technology changes into a more complex form (Henfridsson and Bygstad, 2013).

Routine theory describes the dynamic evolution of routines through the performance of preceding routines. As outlined above, humans do not always perform routines in the same way, which implies that routines change over time or new routines emerge. Humans play a central role in changing existing and generating new routines (Feldman, 2000; Howard-Grenville, 2005). Performing routines thus causes routines to change or new routines to emerge (Pentland et al., 2012). Dittrich and Seidl (2018) argue that means can be created while performing routines. These means can be reused to define and achieve current and new ends. Reusing means in any subsequent performance of a routine results in the routine to change over time (Dittrich and Seidl, 2018). To draw the analogy to the implementation of software robots, one could argue that in the course of programming software robots means are generated that can subsequentially be reused in the further programming of software robots. Such means could be described as components that can be used to build and implement a software robot.

Today, standard robotic implementation solutions are available that provide a kind of toolbox, whereby the robot can be built with the help of the elements contained therein. An example of this is Blue Prism,[1] which allows one to build RPA robots by modeling business processes in processes and objects. A process describes the logic of how an RPA robot should perform a specific task. An object describes the RPA robot's interaction with specific systems on their user interface.

Another example is IBM Watson Conversation Services,[2] which make it possible to create a chatbot by modeling conversations in decision trees and introducing variations and synonyms. A decision tree refers to the structure of a specific dialogue. Variations refer to different semantic structures of questions and answers within the decision tree. Synonyms refer to different terms for the same elements within specific questions and answers.

With regard to RPA robots, means or components that can be reused could be process structures and objects. With regard to chatbots, means or components that can

1 https://www.blueprism.com/

2 https://www.ibm.com/watson/ph-en/conversational-ai/

be reused could be decision trees, variations, and synonyms. Reusing components allows the transfer of routines from humans to robots to be performed more efficiently. In other words, the robot implementation process can be scaled. This essentially implies that the process of transferring routines from humans to robots may have been very complex in the past but certain components have been created that can now be reused. By reusing these same components, the further transfer of routines to robots can be made more efficient and thus scaled.

Thus, scaling can be described as a generative process, which requires actions taken by actors such as the developer. Such actions can be associated with reuse. Reuse enables the development and implementation of IT systems in a more efficient way. Thus, scaling requires that certain components can be reused. Hence, what to scale refers to the components that can be reused in the further implementation of software robots.

6.2.2 How to scale

Reuse of already created components can be considered as a mechanism that triggers scaling by enabling the addition or transfer of functionality (Banker and Kauffman, 1992; Basili et al., 1996). The flexibility of technology can thereby be innovatively exploited by extending functionality within the same or a new environment or context. Thus, the addition of new functionalities (mutation) to an IS can describe one mode of scaling. Another mode of scaling describes the transfer of functionalities (inheritance) to a new IS (Huang et al., 2017; Svahn et al., 2017; Yoo et al., 2012). In regard of the implementation of software robots this means that components may be reused in the same or a new environment or context. Hence, how to scale refers to whether components are reused to transfer functionality into a different or new context or to extend functionality in a current context.

6.2.3 To what extent to scale

However, not everything can be reused directly but certain components may first have to be modified so that they can then be reused to transfer or extend functionality. The reuse mechanism can therefore not always be applied directly but depends on contextual factors (see Adler et al., 1999). It is important to understand what the contextual factors are that prevent direct reuse and how components can be adapted in order to be reused (see Adler et al., 1999).

If components can be reused directly, it can also be said that they can be reproduced into an existing context. If means cannot be directly reused, it can also be said that they must be recreated into a different or new context. Reproduction indicates

that means can be directly reused regardless of contextual factors. Recreation indicates that means must be adapted depending on contextual factors in order to be reused (Dittrich and Seidl, 2018).

Hence, to what extent to scale refers to whether components need to be adapted or not in order to be reused within a certain context and, if so, how components need to be adapted.

6.3 Conceptualization of scaling the implementation of software robots

Thus, in order to better understand the scaling of implementing software robots, it is necessary to analyze which components can be reused (what), how, and to what extent (Henfridsson and Bygstad, 2013; Huang et al., 2017; Svahn et al., 2017; Yoo et al., 2012). Implementing software robots by transforming human-executed processes (i. e. routines) into robot-automated processes can be done more efficiently as the implementation process scales. Based on theoretical concepts of routine theory, as well as digital scaling literature and our preliminary data, we have developed an initial model of scaling the implementation of software robots (see Figure 1). Routine theory and digital scaling literature allow us to theoretically open the black box of a successful evolution of software robots and how the associated scaling can be approached.

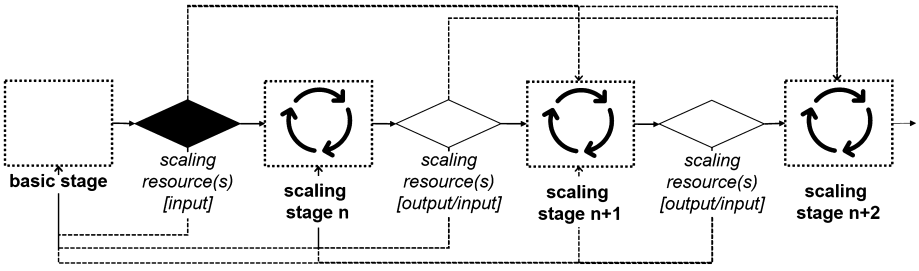

Figure 1: Phase of scaling.

The model is composed of multiple elements (see Table 1). Here, scaling the implementation of software robots is described as a circular process in the context of which scaling resources are created that can be reused in the further course of scaling. Thus, scaling resources refer to components that can be created in the course of implementing or programming software robots. Once created, scaling resources can be reused to accelerate or scale the implementation of software robots.

The whole scaling process can be divided into different stages of scaling, which can take place sequentially or in parallel. Scaling can be initiated once a basic stage

Table 1: Description of scaling elements.

	Description
Basic stage	The basic stage describes a first phase of the development of a software robot, based on which the robot implementation can be scaled subsequently
Scaling resource(s)	Scaling resources describe components that were created in the course of the previous development of a software robot and can be reused in further development. They can result as output from a scaling stage and they can be used as input to continue in a current scaling stage or to initiate a new scaling stage
Scaling stage	A scaling stage describes a phase of scaling in the implementation of software robots

has been completed. The basic stage refers to a first phase of the development of a software robot. Thus, the initial development or programming of the software robot can not yet be scaled but a software robot must already have been developed to a certain degree so that its further development can be scaled. Scaling resources can result from the basic stage and from each subsequent scaling stage. They can be considered as necessary input to enable scaling in each scaling stage and as potential output that can result from the basic stage and each subsequent scaling stage. As a resulting output, scaling resources can be reused in any past, current, or subsequent scaling stage. Thus, scaling resources can be reused not only prospectively but also retroactively either in the scaling stage from which they resulted or in any preceding scaling stage.

6.4 Illustration through case data

To better understand scaling in the context of implementing software robots, we have made first steps towards a theoretical understanding and empirical illustration of the scaling of the implementation of software robots. We have chosen a reciprocal approach that has enabled us to conceptualize a model, in which we have derived theoretical elements of routine theory and digital scaling literature deductively and inductively from empirical data of two case studies. However, the model developed in this chapter represents a first draft and needs to be further refined and substantiated with additional data. With the help of theoretical sampling, we identified two cases that seemed to contribute to an empirical illustration of our conceptualized model. Case 1 describes a chatbot project at a Swiss bank. Case 2 describes an RPA project at another Swiss bank. The aim is not to test the model but to illustrate it and to show how our model can be instantiated while some aspects of the model have also been derived from the case data (see, e. g., Leonardi, 2011). We conducted nine semi-structured interviews with people in different roles within the project team of case 1 between Oc-

tober and November 2017, in a second round in September 2018, and in a third round in March and April 2020. For case 2, we conducted three semi-structured interviews in October and November 2017. This helped us to obtain a holistic picture of both cases (Miles and Huberman, 1994; Yin, 2003). In addition, we also analyzed other data such as robot software suit manuals. Given that our key objective is to build theory, the research thrust is exploratory in nature (Benbasat et al., 1987). Qualitative research methods are suitable for "generating novel theory" (Eisenhardt, 1989, p. 546) – in particular theory that aims at answering "how" and "why" questions (Yin, 2003). This is true for our study as the key objective is to understand how to scale the robot implementation process.

6.4.1 Case narrative 1

The case describes a bank that wanted to optimize its contact center (CC) in terms of efficiency and in terms of improving performance and reducing costs. Therefore, chatbots were deemed suitable to automate processes. The project regarded was initiated in October 2016.

The team consisted of a product owner, a scrum master, an external partner, an application manager, and the content team. The product owner and the scrum master were primarily responsible for managing the project. The product owner focused mainly on the business aspects and the scrum master on the IT aspects of the project. The external partner helped the bank to acquire the necessary know-how to implement the chatbot. Since it was the first chatbot project of the bank, this was essential. The application manager was responsible for maintaining the chatbot application and integrating it into the existing system architecture. The role of the content team was to generate the content, based on which the chatbot should conduct dialogues. By content, topics and corresponding dialogues are meant. The content team therefore had to identify such topics on the one hand and then define and implement corresponding dialogues. An example of such a topic would be the query of the current account balance by a customer.

The chatbot software used by the bank was relatively simple. Decision trees could be modeled graphically. Thus, the content team mostly modeled dialogues directly in the chatbot framework itself. The chatbot framework describes the chatbot software used by the bank and its functionalities, which the project team used to build the chatbot.

In the course of the project, the chatbot was not only continuously developed but the project team was able to benefit from already created components to the extent that it was able to scale the entire implementation process of the chatbot. Thus, by evaluating the project, different scaling stages could be identified. In summary, three different incidents, which each reflect one stage of scaling could be identified that together

represent an evolutionary path that the chatbot went through during its implementation. These are the scaling from the German to the French chatbot, the scaling to the e-banking chatbot, and the scaling to the voicebot. In each of the three scaling stages certain components could be reused and thus functionalities could be transferred. It was, however, only possible to digitally scale after the chatbot had been developed and implemented to a certain extent, i. e., the German bot had been created.

6.4.1.1 Basic stage – German bot

The basic prerequisite for scaling the implementation of the bank's chatbot was therefore the previous development and implementation process of the basic stage of the chatbot, i. e., the German version of the chatbot. The development of the basic stage basically meant to model German conversational processes within decision trees and to implement variations and synonyms. Decision trees were modeled around one main question, which constituted the root, while possible direct answers and follow-up questions formed the branches of each decision tree. One main question then required about 100 variations, so that the chatbot was able to answer accurately. "Still, if there is a 101st question and the syntax is wrong, we are pretty sure the chatbot is going to map the question to the right main question" (external partner). Decision trees refer to a dialogue's structure, while the main questions refer to dialogue topics.

Initially, the chatbot could answer simple questions in German that contained general information; occurred in high volumes; contained self-service components or aspects that the end user could handle him or herself; and referred to a non-value adding process. Thus, in this basic stage the chatbot was able to conduct simple conversations in German, which did not require any kind of system integration. Users could access the chatbot through the bank's website without being logged in.

As long as decision trees were extended and new variations and synonyms were added that allowed the chatbot to run processes more accurately in the same domain, i. e., around the same topic, no scaling was performed. Thus, the initial development and implementation phase of the chatbot cannot be referred to as scaling. The German version of the chatbot was officially released in November 2017.

6.4.1.2 Scaling stage 1 – French bot

After completion of the basic stage, in which the German chatbot was developed, the first scaling stage of the development of the French chatbot could be initiated. Based on the basic stage the project team could further develop the chatbot so that it could conduct conversations not merely in German but also in French. This can be referred to as the first scaling stage. The basic idea of this stage was to build on the conversations or dialogue topics that had previously been built up in German. These conversations

should be translated into French. As the decision trees of the German dialogues already contained a considerable amount of questions and corresponding answers, the project team assumed that this stage could be completed relatively quickly. In principle, the project team assumed that the dialogues previously set up in German could now be translated directly into French. However, the complete reuse of dialogue topics and dialogue structures was not possible as only some components could indeed be reused. Reuse was prevented by the users who were supposed to use the chatbot in French. The reason for this was that the French speaking users expressed themselves and structured dialogues differently than the German speaking users.

During the development of the German version of the chatbot, the project team acquired considerable knowledge about the extent to which a chatbot needs to be developed at all, and which topics the users want to address via the chatbot. The project team could build on this knowledge to develop the French version of the chatbot faster. In addition, however, they had to analyze and better understand their French speaking users as well as their behaviors in order to set up and structure the French dialogues accordingly. To accomplish this, the content team has been expanded to include a native French speaker.

Thus, in the first scaling stage, some components could be directly reused, while for the reuse of other components, these first had to be made compatible with the corresponding context. Concretely, this meant that the knowledge acquired for the development of the chatbot and the dialogue topics previously established for the German bot could be directly reused for the French bot. The reuse of the knowledge enabled the project team to design the dialogs for the French bot more efficiently and faster. The German dialogue structures, on the other hand, could not be directly reused, although their structure had to be revised to suit the French speaking users. The French chatbot was released in October 2018.

6.4.1.3 Scaling stage 2 – e-banking bot

The second scaling stage then describes the evolution from the development of the German and French dialogues to the integration of the chatbot into the e-banking system. This should allow the users of the chatbot not only to have general conversations but also to ask user-specific questions while being logged into the bank's e-banking system. Until then, the chatbot was dependent on the information the user provided in a chat. With the integration into the e-banking system, the chatbot could now directly retrieve information about a specific user from the system. However, this not only meant that the conversations needed to be tailored to a specific user but it also meant that sensitive user-specific data should become part of the dialogues.

The chatbot software used by the bank was based on cloud servers located abroad. However, the Swiss data protection law stipulated that sensitive user data may only be processed on cloud servers located in Switzerland. Consequently, for the bank

this meant that the dialogues processed on the cloud servers of the chatbot software provider could not contain any sensitive data. The project team was confronted with the problem that with the integration into the e-banking system, sensitive data would become part of the conversations but that these sensitive data cannot be sent via the chat, because otherwise it would be stored in the chatlog on the cloud abroad.

The project team solved this problem with so-called deep links. In doing so, the user could for example ask, "What is my account balance?" Rather than the chatbot answering the user's account balance in the chat field, the chatbot would reply "You can find your account balance in the red marked field on your screen." If it was sensitive data that the chatbot was supposed to give out, it was not displayed in the chat field but highlighted directly on the bank's website. The sensitive data entered by users were encrypted in order to ensure compliance with data protection laws.

For the integration of the chatbot into the e-banking system, the project team was once again able to draw on the knowledge they had already gained in developing the German and French bots. In addition, the previously modeled dialogue topics in German and French could be reused. These dialogue topics were previously modeled in a very general and not user-specific manner. The integration into the e-banking system, however, should now focus on the chatbot being able to conduct user-specific dialogues. The previous dialogue topics could thus be used as a basis, which now had to be tailored specifically to the user.

In addition to the already existing dialogue topics, new topics had to be covered by new dialogues. When modeling these new dialogue topics, certain dialogue structures of the previous dialogues could be reused.

In the second scaling stage, most components could be directly reused. The reuse of these components enabled the project team to design the dialogues for the e-banking bot more efficiently and faster. The integration into the e-banking system was initiated in summer 2018 and a first version got released in early 2019.

6.4.1.4 Scaling stage 3 – voicebot

Finally, the third scaling stage describes the evolution from the integration into the e-banking system to the extension of the chatbot to a voicebot. This third stage was initiated shortly after the start of the second stage of the integration into the e-banking system. The voicebot should allow users to interact with the bot not merely by text input but also by voice input. The users were to reach the voicebot by phone, just as they had reached CC employees before. In a first phase, no integration of the voicebot into the web site or the mobile application was planned. The project team still left this possibility open.

The project team again assumed that a considerable part of the dialogue topics and structures already created for the chatbot could be reused for the development of the voicebot. This was again not as easy as one had hoped for. The project team

had to realize that not only the German and French speaking users expressed themselves differently, but all users spoke differently than they wrote. "For example, the syntax is completely different when the customer asks, 'Can I check my account balance, please?' Then he writes on the text channel: 'Account balance please'. Maybe two words. [. . .] But when he enters it in the voice channel, it's more of a dialogue and he says, 'Yes, I think I got my paycheck yesterday and I need to know what my balance is and check if I can pay my bills.' [. . .] And you just can't compare how the customers write and how they talk to the assistant [voicebot]" (product owner).

Thus, it turned out to be much more difficult to reuse already modeled dialogues for the voicebot. The project team therefore had to rethink its approach. They did this by adopting a voice-first approach and trained all team members accordingly. Voice-first meant that the project team would create all newly generated dialogues for the voice channel first, in a form in which they could potentially be used for the text channel in a second step.

The project team also reworked some of the existing dialogue topics and structures so that they were compatible for the voice channel. "We will not be able to make 100 % of the content we have modeled suitable for voice. That would lead to too much effort at the moment" (product owner).

As with the integration into the e-banking system, the transition to voice posed the challenge of sensitive data. However, this time it was not possible to use deep links and show the sensitive aspects of a conversation to the respective user on the visual user interface. Once conversations are conducted via voice channel, there is no visible user interface. As mentioned above, the bank was not allowed to process sensitive data on a foreign cloud due to applicable regulations. The provider of the chatbot software used until then was neither willing to install a cloud in Switzerland, nor to offer the software on-premise. The bank was thus forced to look for another solution.

They found what they were looking for in another provider who delivered their chatbot software on-premise. The text-based chatbots should run on the old chatbot software for the time being. The voicebot should be based on the new chatbot software. However, the long-term goal was to completely replace the old chatbot software with the new one.

For the development of the voicebot the project team was not able to directly reuse the dialogue topics and structures that have been created in previous stages. Instead, the dialogue structures had to be made compatible with the voice channel. Accordingly, the project team translated some of the text dialogues into voice dialogues. However, some of the text dialogues hardly seemed suitable for the voice channel. Here the project team had to reverse the approach and henceforth model voice-first dialogues, which could then be reused for the text channel and thus the further extension of the text-based bots.

In the third scaling stage, most components could not be directly reused but they had to be revised to suit the voicebot. Reuse was reversed here in that the dialogue structures created for the voicebot were to be reused retroactively for the text-based

bots. The transition from the text to the voicebot is still in progress. A first version of the voicebot was released in June 2020.

6.4.2 Case narrative 2

The case describes another bank that also wanted to optimize its CC in terms of efficiency, improved performance, and reduced costs. Therefore, multiple RPA robots should allow the automation of business processes. The project was initiated in July 2017. The project team consisted of different roles. As this was the bank's first RPA project, the project was implemented in collaboration with an external consultancy firm.

Similarly to the case previously outlined, certain scaling dynamics could be identified in this case as well. Unlike in the chatbot case, however, no multiple scaling stages could be identified but scaling was instead applied step by step within one single stage of scaling. In this scaling stage, various business processes were to be automated with the help of several RPA robots.

6.4.2.1 Basic stage

The basic prerequisite for scaling the implementation of the bank's RPA robots was an initial phase where the project team had to understand the RPA robot design. This was critical, because it determined how business processes could be introduced to the RPA software so that an RPA robot could execute them.

The RPA software used allowed the programming of RPA robots that could perform a sequence of process steps and mimicking what the human user normally does. The automation of business processes through the development of RPA robots was thereby done in the software's Studio, which was divided into Process Studio and Object Studio. Process Studio enabled the configuration of the process logic and the business rules (i. e., the process structure). Object Studio enabled the creation of reusable objects. A process described the logic of how a specific RPA robot executed tasks. An object described the RPA robot's interaction with specific systems on their user interface.

The developers did not actually have to program the automation of business processes but could graphically model them with the help of various flowchart elements. In Process Studio, one could either entirely model business processes or split them into multiple process steps. Each process step could be modeled in a separate page. Throughout all the pages, the main process could be kept slim on the main process page; frequently used process steps within a particular process could be reused.

In Object Studio objects could be created, which allowed integrating external systems into the RPA software framework. With the "spying mode" of Object Studio, ev-

ery system button could be tracked and added to the corresponding object. Once a system and its entire corresponding buttons had been integrated, actions linked to the usage of a specific system could be modeled. Unlike in Process Studio, pages were hereby used to model individual actions related to a specific object. For example, in one of the business processes to be automated, the RPA robot had to send a confirmation letter to a customer who had opened a new account. For this purpose, the RPA robot had to know the respective system button "print" and execute the action "print confirmation letter." To then add an action to a process in Process Studio, one could access the corresponding action from Object Studio. To do this, the flowchart element "action" had to be inserted into the main process or a process step page in Process Studio.

In summary, Object Studio enabled the integration of specific systems needed so that the RPA robots could execute the business processes modeled in Process Studio. Once the project team had gained an understanding of the design of the RPA software, the actual process automation could be initiated.

Business processes suitable for automation had to be executed in high volume and on a computer; they had to be rule-based and should entail limited exceptions; they should implicate structured data, and each business process to be automated should replace 0.3 full-time equivalents (FTE) in order to achieve the break-even point after one year. It was only worthwhile to develop a robot in case it could undertake the work of 0.3 FTE. Based on these criteria and a list of all processes executed in the CC, the project team identified four business processes with automation potential. Those four processes should be automated first while potential additional processes should follow later.

The development of the first four RPA robots was initiated with the modeling of the respective business processes. This was done within Process Studio and Object Studio. Each RPA robot was hereby set up through one process containing various objects that described the actions an RPA robot had to take in various process steps. However, some of the developers initially created RPA robots within objects instead of forming processes by using objects. "The object is something that you can reuse. The process is something you are only using for the current robotic process. So… you should not create a process inside an object. But many times, they did it" (supplier chief developer). If done so, objects could only be used for one specific process, while reuse was not possible. However, the idea of using objects to build processes was to be able to reuse the objects for several processes involving the same systems and thus to scale the RPA robot implementation process. Even though this approach required more effort in the beginning, it allowed a more efficient and faster implementation of subsequent RPA robots accessing the same systems. "Because the first robots are always the hardest. How so? Because… you develop that in objects. These are objects that can be reused in other robots. This automatically means that subsequent robots can be developed faster" (supplier project manager 2). Once the chief developer discovered that the other developers defined processes within objects instead of using

various objects to define one process, he drew their attention to it, and they changed their approach from object-based to process-based development.

The initial development and implementation phase of the RPA robots cannot be referred to as scaling but allowed the creation of objects that could be reused later on.

6.4.2.2 Scaling stage 1

After completion of the basic stage, in which initial objects were created, scaling could be initiated. The automation of the four processes originally identified took place in parallel. The basic stage was not completed with the completion of the implementation of the four corresponding RPA robots. Rather, it was possible to move from the basic stage to scaling stage 1 after some initial objects were created that could be reused continuously and thus contributed to implement RPA robots more efficient. Thus, the previously created objects could be reused in the further implementation of the RPA robots. This helped to speed up the overall implementation process of the robots.

In the course of the development of multiple RPA robots some components, i. e., objects and process structures, could be reused. This enabled a more efficient implementation of the RPA robots and thus scaling. However, not only were existing objects and process structures reused but also new ones were created even after the completion of the basic stage. Scaling was thus achieved through the creation of objects and process structures. After a period of five months, the first out of the initial four RPA robots was released in November 2017.

6.5 Discussion

Implementing software robots essentially describes an approach of automating business processes. In this chapter we associate such business processes with routines. The performance of routines often implies that certain means are used to achieve certain ends. It has been argued in previous literature that the ends determine which means should be used in achieving ends (Feldman et al., 2016). Dittrich and Seidl (2018) add to this that certain means originate in the performance of routines and can be used to define and achieve new ends. A similar dynamic can be observed in the implementation of software robots.

One could describe the implementation of software robots as automating processes by means of programming software robots (Rutschi and Dibbern, 2019, 2020). While implementing software robots, components or resources can be created that can be reused for the further programming of past, current, or new robots. This allows for a more efficient and faster programming of software robots and thus to accelerate or scale the software robot implementation process (Dittrich and Seidl, 2018).

By scaling the implementation of software robots, functionality of a current robot can be extended or transferred to a different robot or context. As a result, the evolution of software robots can be accelerated. Such acceleration or scaling can be achieved when certain resources are created during the initial programming of software robots. These same resources can then be reused.

To better understand how to scale the implementation of software robots, it is essential to understand which resources can be reused, how they can be reused (in the same or a different context), and to what extent they can be reused (through reproduction or recreation) (Henfridsson and Bygstad, 2013; Huang et al., 2017; Svahn et al., 2017; Yoo et al., 2012). In this chapter we provide insights into all three of these aspects: (1) what to scale (i. e., which means or resources), (2) how to scale (i. e., through which type of reuse), and (3) to what extent to scale (i. e., the type of contextual extension). In the following, we will again discuss and elaborate on all three aspects and show how they apply in the two cases analyzed.

6.5.1 What to scale

In order to scale at all, certain resources or means must be created, which can be reused subsequently. Hereafter we will go into more detail about what such resources could look like, and which concrete resources were created in the two cases. As outlined above, a routine consists of an ostensive and a performative aspect (D'Adderio, 2011; Feldman et al., 2016; Pentland and Feldman, 2005). If we now want to transfer a routine from a human to a robot, we need to comprehend both the ostensive aspect and the performative aspect of the human-executed routine. Only then can we transfer the human-executed routine into a robot-automated routine. Therefore, we need to translate the ostensive and performative aspects of the human-executed routine into ostensive and performative aspects of the robot-automated routine.

The ostensive and performative aspects differ from each other when a routine is performed by a human, since the human can continuously influence both aspects. This is different for software robots. The robot has an initial influence on the ostensive aspect of the routine by specifying the extent to which it can and cannot perform a routine. The rules and procedures (i. e., the ostensive aspect) according to which a human has executed the routine in the past must therefore be translated to correspond to the robot design.

Once a routine has been implemented as a software robot, it does not change anymore. The robot performs the routine by strictly following the given rules and procedures (i. e., the ostensive aspect). Different from when a human performs a routine, the ostensive and performative aspects are the same when a routine is executed by a software robot.

When a company decides to implement software robots to automate certain business processes, there are several ways to approach this. The company can automate

each process individually, starting from scratch each time. Alternatively, the company can combine the automation of several processes and build on already created resources or means. In fact, these resources can be referred to as the ostensive aspect of the routine or a part of it, which is implemented as a robot.

For each automation of a process Pn+1, which is preceded by another automation of a similar process Pn, it must be examined to what extent the ostensive aspect On of process Pn corresponds to the ostensive aspect On+1 of process Pn+1. The difference or delta between On and On+1 must be evaluated (see Figure 2). The more the two overlap and the larger the delta, the more resources can be reused. Therefore, the delta describes what can be reused or scaled. If the delta is positive, scaling resources can be reused in any current or subsequent scaling stage. If the delta is negative, scaling resources can be reused retroactively.

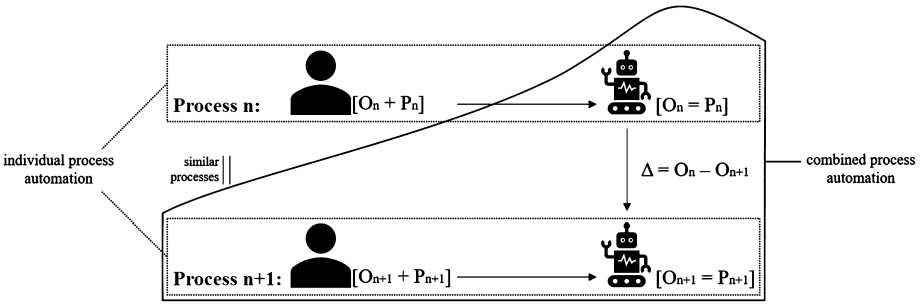

Figure 2: Individual and combined process automation.

In case 1, the implementation of the German, French, and e-banking bots and the voicebot led to the generation of scaling resources (i. e., means) such as dialogue topics (in German and French) and dialogue structures (in German and French, and of general as well as of user-specifically tailored text- and voice-based nature). Reusing these scaling resources resulted in an acceleration of the implementation of the chatbot (see Table 2).

In case 2, the implementation of the first RPA robots led to the generation of scaling resources (i. e., means) such as process structures and objects. Reusing these scaling resources resulted in an acceleration of the implementation of the RPA robots. However, the project team did not initially create scaling resources but some of the developers created RPA robots within objects instead of forming processes by using objects. As a result, every new development of an RPA robot had to be started from scratch and could not be based on already existing components or scaling resources. When they realized this, they adapted their implementation approach and created components (i. e., scaling resources) that could be reused (see Table 3).

Table 2: Scaling process in case 1.

Basic stage	Scaling resource(s) (input)	Scaling stage 1	Scaling resource(s) (output/ input)	Scaling stage 2	Scaling resource(s) (output/ input)	Scaling stage 3	Scaling resource(s) (output/ input)
Development of German bot	Dialogue structures and dialogue topics in German	Development of French bot	Dialogue structures and dialogue topics in French	Development of e-banking bot	Dialogue structures, additional dialogue topics, and deep link integrations	Development of voicebot	Voice-first dialogue structures and additional dialogue topics

Table 3: Scaling process in case 2.

Basic stage	Scaling resource(s) (input)	Scaling stage 1	Scaling resource(s) (output/input)
Initial development of RPA robots	Process structures and objects	Further development of RPA robots	Additional process structures and objects

6.5.2 How to scale

Once it has been defined what (i. e., what scaling resources) can be scaled, one must evaluate how to scale. Scaling resources can be reused to accelerate the implementation of a current, a subsequent, or a previous software robot. Scaling resources can be reused to extend functionality of a robot or to transfer functionality from one robot to another. This results in two types of reuse (i. e., how to scale). Expanding functionality can be referred to as *mutation*. Transferring functionality can be referred to as *inheritance* (Henfridsson and Bygstad, 2013).

In case 1, scaling resources were reused for extending as well as transferring functionality. Scaling resources that resulted from the basic stage (i. e., implementing the German bot) were reused to accelerate the further development of the German bot (first context) as well as the initial developments of the French bot (second context), the e-banking bot (third context), and the voicebot (fourth context). Scaling resources that resulted from scaling stage 1 (i. e., implementing the French bot) were reused to accelerate the further developments of the German (first context) and French bots (second context) as well as the initial developments of the e-banking bot (third context) and the voicebot (fourth context). Scaling resources that resulted from scaling stage 2 (i. e., implementing the e-banking bot) were reused to accelerate the further development of the e-banking bot (third context) as well as the initial development of the voicebot (fourth context). Scaling resources that resulted from scaling stage 3 (i. e.,

implementing the voicebot) were reused to accelerate the further development of the voicebot (fourth context) as well as the further development of the German (first context), French (second context), and e-banking bots (third context).

In case 2, scaling resources were reused in order to extend functionality of current RPA robots as well as to transfer functionality to other RPA robots.

6.5.3 To what extent to scale

Finally, one must understand in what context scaling takes place (i. e., the type of contextual extension). Direct reuse may be hampered whereas scaling resources may not always be reproduced but need to be recreated if they are to be reused in a different context. *Reproduction* indicates that means or scaling resources were directly reused independent of the context. *Recreation* indicates that means or scaling resources were adapted according to a new context in order to be reused.

In case 1, some scaling resources (i. e., dialogue topics) could be reused directly while others (i. e., dialogue structures) had to be adapted to different contexts in order to be reused. Overall, scaling dynamics could be observed in multiple (i. e., three) scaling stages within case 1.

In case 2, all RPA robots should be implemented in the same department. Thus, scaling resources (i. e., process structures and objects) could be reused directly without any need for context-specific adaptations in case 2. Unlike in case 1, in case 2 scaling dynamics could merely be observed in one stage of scaling.

6.6 Conclusion and outlook

Software robots are expected to dramatically improve the efficiency of companies and disrupt the way humans and machines work and collaborate (Willcocks and Lacity, 2016). It is crucial for companies to understand how business processes can be automated successfully by implementing software robots and how such robot implementations can be scaled. Implementing software robots by transforming human-executed routines into robot-automated processes can be done more efficiently by scaling the implementation process.

Despite the automation through software robots, the human factor plays an important role here. This may change as technology advances and robots are able to learn how to take over and perform certain processes autonomously without any need for human intervention. Given the type of software robots (i. e., chatbots and RPA robots) we are looking at here, this is not the case. Such software robots can take over and perform processes only after a human (i. e., the developer) has programmed the robot accordingly.

For this to be done more efficiently, the human must understand the delta and thus the degree of scaling. In other words, the developer must understand the extent to which the ostensive aspect of two similar processes or routines to be automated overlap. Depending on whether the delta between two ostensive aspects is positive or negative, scaling resources can be reused for current and subsequent robot implementations or retroactively for preceding robot implementations.

We contribute to routine theory and literature on digital scaling by examining how the implementation of software robots can be scaled. Specifically, we found that (1) for scaling the implementation of software robots, one can build on what already exists (scaling resources), (2) scaling resources can be reused for functionality extension or transfer (mutation vs. inheritance), and (3) scaling resources can be reused in different contexts, in which they can be reused directly or through adaptations (reproduction vs. recreation).

In this regard, we have conceptualized a model for scaling the implementation of software robots based on existing constructs from routine theory and digital scaling literature. Some aspects of the model have also been derived from the data.

Besides the implications of our research, we also must acknowledge its limitations. The model developed in this chapter describes an initial model and needs to be further refined and substantiated with additional data.

We have shown exemplarily how scaling was approached in two cases. It was shown that digital scaling can be divided into different scaling stages, within which scaling resources (i. e., means) are created and can be reused. The reuse of scaling resources can be considered as a mechanism that triggers scaling by enabling the extension (i. e., mutation) or transfer (i. e., inheritance) of functionality (Banker and Kauffman, 1992; Basili et al., 1996). The implementation of software robots is associated with high costs and time expenditure. These can be reduced by scaling and therefore the implementation of software robots can be made more efficient. However, in order to be able to scale at all, it must be understood what can be scaled (i. e., what scaling resources or means), how it can be scaled (i. e., in what contexts), and to what extent it can be scaled (i. e., through reproduction or recreation).

An understanding of how to scale the software robot implementation process is of great interest to both research and practice. We make first steps in conceptualizing and theorizing the three aspects (i. e., what, how, and to what extent) of scaling the implementation of software robots. Future research could seek to better understand which contextual factors could impede direct reuse of scaling resources and why reuse is performed this or that way.

Bibliography

Adler PS, Goldoftas B, Levine DI (1999) Flexibility versus efficiency? A case study of model changeovers in the toyota production system. Organ Sci 10(1):43–68

Asatiani A, Penttinen E (2016) Turning robotic process automation into commercial success – case OpusCapita. J Inf Technol Teaching Cases 6(2):67–74

Banker RD, Kauffman RJ (1992) Reuse and productivity in integrated computer-aided software engineering: an empirical study. MIS Q 14(3):374–401

Basili VR, Briand LC, Melo WL (1996) How reuse influences productivity in object-oriented systems. Commun ACM 39(10):104–116

Benbasat I, Goldstein DK, Mead M (1987) The case research strategy in studies of information systems. MIS Q 11(3):369–386

Chandler AD (1990) Strategy and structure: chapters in the history of the industrial enterprise, vol 461. MIT press

D'Adderio L (2011) Artifacts at the centre of routines: performing the material turn in routines theory. J Inst Econ 7(2):197–230

Dittrich K, Seidl D (2018) Emerging intentionality in routine dynamics: a pragmatist view. Acad Manag J 61(1):111–138

Eisenhardt KM (1989) Building theories from case study research. Acad Manag Rev 14(4):532–550

Feldman MS (2000) Organizational routines as a source of continuous change. Organ Sci 11(6):611–629

Feldman MS, Pentland BT (2003) Reconceptualizing organizational routines as a source of flexibility and change. Adm Sci Q 48(1):94–118

Feldman MS, Pentland BT, D'Adderio L, Lazaric N (2016) Beyond routines as things: introduction to the special issue on routine dynamics. Organ Sci 27(3):505–513

Fung HP (2014) Criteria, use cases and effects of information technology process automation (ITPA). In: Advances in robotics & automation, vol 3

Guzman I, Pathania A (2016). Chatbots in customer service. Retrieved 25 Oct, 2017, from https://www.accenture.com/t00010101T000000__w__/br-pt/_acnmedia/PDF45/Accenture-Chatbots-Customer-Service.pdf

Heller B, Proctor M, Mah D, Jewell L, Cheung B (2005) Freudbot: an investigation of chatbot technology in distance education. Paper presented at the EdMedia: World Conference on Educational Media and Technology

Henfridsson O, Bygstad B (2013) The generative mechanisms of digital infrastructure evolution. MIS Q 907–931

Howard-Grenville JA (2005) The persistence of flexible organizational routines: the role of agency and organizational context. Organ Sci 16(6):618–636

Huang J, Henfridsson O, Liu MJ, Newell S (2017) Growing on steroids: rapidly scaling the user base of digital ventures through digital innovation. MIS Q 41(1)

Leonardi PM (2011) When flexible routines meet flexible technologies: affordance, constraint, and the imbrication of human and material agencies. MIS Q 35(1):147–167

Miles MB, Huberman AM (1994) Qualitative data analysis: an expanded sourcebook. Sage Publications, Thousand Oaks, CA, USA

Patil A, Marimuthu K, Niranchana R (2017) Comparative study of cloud platforms to develop a chatbot. Int J Eng Technol 6(3):57–61

Pentland BT, Feldman MS (2005) Organizational routines as a unit of analysis. Ind Corp Change 14(5):793–815

Pentland BT, Feldman MS, Becker MC, Liu P (2012) Dynamics of organizational routines: a generative model. J Manag Stud 49(8):1484–1508

Pfeifer R, Lungarella M, Iida F (2007) Self-organization, embodiment, and biologically inspired robotics. Science 318(5853):1088–1093

Rutschi C, Dibbern J (2019) Mastering software robot development projects: understanding the association between system attributes & design practices. In: Paper presented at the proceedings of the 52nd Hawaii international conference on system sciences, Hawaii, USA

Rutschi C, Dibbern J (2020) Towards a framework of implementing software robots: transforming human-executed routines into machines. ACM SIGMIS Database 51(1):104–128

Sahay S, Walsham G (2006) Scaling of health information systems in India: challenges and approaches. Inf Technol Dev 12(3):185–200

Sengupta R, Lakshman S (2017) Conversational chatbots – let's chat. Retrieved 25 Oct, 2017 from https://www2.deloitte.com/content/dam/Deloitte/in/Documents/strategy/instrategy-innovation-conversational-chatbots-lets-chat-final-report-noexp.pdf

Sharma P, Southern R, Dalton D (2016) The discruptive chat bots – sizing up real opportunities for business. Retrieved from https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/ie-dispruptivechat-bots.pdf

Shawar BA, Atwell E (2007) Different measurements metrics to evaluate a chatbot system. In: Paper presented at the proceedings of the workshop on bridging the gap: academic and industrial research in dialog technologies, Rochester, NY, USA

Slaby JR (2012) Robotic automation emerges as a threat to traditional low-cost outsourcing. HfS Research Ltd.

Svahn F, Mathiassen L, Lindgren R (2017) Embracing digital innovation in incumbent firms: how volvo cars managed competing concerns. MIS Q 41(1)

Tirgul CS, Naik MR (2016) Artificial intelligence and robotics. Int J Adv Res Comput Eng Technol 5(6):1787–1793

Willcocks L, Lacity MC (2016) Service automation: robots and the future of work. Steve Brookes Publishing

Yin RK (2003) Case study research. In: Applied social research methods series, vol 5. Sage Publications, Beverly Hills, CA, USA

Yoo Y, Boland RJ Jr, Lyytinen K, Majchrzak A (2012) Organizing for innovation in the digitized world. Organ Sci 23(5):1398–1408